

A084022

LEVEL 1

NSWC TR 80-18

# PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION FOR THE AN/UYK-7 CENTRAL PROCESSOR

by

ALAN HYNSON  
JAY MEYERS  
CHARLES NAPLES  
Strategic Systems Department



FEBRUARY 1980

Approved for public release; distribution unlimited.



NAVAL SURFACE WEAPONS CENTER

Dahlgren, Virginia 22448

Silver Spring, Maryland 20910

JMC FILE COPY

80 5 8 008

**NAVAL SURFACE WEAPONS CENTER  
Dahlgren, Virginia 22448**

**Paul L. Anderson, Capt., USN  
Commander**

## UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC/TR-84-18	2. GOVT ACCESSION NO. AD-A084 021	3. RECIPIENT'S CATALOG NUMBER
PROGRAM DESIGN LANGUAGE ARCHITECTURE SPECIFICATION OF THE AN/UYK-7 CENTRAL PROCESSOR <i>for Central</i>		4. TYPE OF REPORT & PERIOD COVERED Final Report
5. AUTHOR(S) Alan J. Hynson Jay C. Meyers Charles J. Naples	6. CONTRACT OR GRANT NUMBER(S) 12 3001	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (K74) Dahlgren, VA 22448	8. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NIF	
9. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Weapons Center (K74) Dahlgren, VA 22448	10. REPORT DATE February 1980	11. NUMBER OF PAGES 203
12. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. SECURITY CLASS. (of this report) Unclassified	
14. DECLASSIFICATION/DOWNGRADING SCHEDULE		
15. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
16. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
17. SUPPLEMENTARY NOTES		
18. KEY WORDS (Continue on reverse side if necessary and identify by block number) Program Design Language (PDL); emulation, computer architecture, microprogramming, AN/UYK-7, central processor (CP)		
19. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Univac AN/UYK-7 computer is the Navy's standard mainframe computer for medium and large scale applications. This report presents a Program Design Language (PDL) description of the AN/UYK-7 Central Processor (CP) architecture and is intended for use as a reference document.		
20. This PDL was used as a design specification for a microprogrammed emulation of the AN/UYK-7 Central Processor. This emulation has been (continued)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 68 IS OBSOLETE  
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411567 11

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

validated through the successful execution of the ~~univer-~~ supplied AN/UYK-7  
Central Processor (CP) diagnostics and by an independent testing effort.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This document contains a comprehensive description of the central processor architecture of the Navy Standard AN/UYK-7 computer. The architecture was described using a Program Design Language (PDL) supplied by Caine, Farber, and Gordon, Inc. As far as can be determined this document contains the most accurate description of the central processor of the AN/UYK-7 computer published to date.

The PDL was used in the development of a microprogrammed AN/UYK-7 central processor emulation on a Nanodata QM-1 computer. This emulation has been validated through the successful execution of the UNIVAC-supplied AN/UYK-7 CP diagnostics and by an independent testing effort.

The authors gratefully acknowledge the efforts of Henry Walker (SPERRY UNIVAC) and Marc Hubbard (Gun Fire Control Systems Branch, Combat Systems Department) for providing assistance in clarifying ambiguous UNIVAC documentation and verification of architectural questions on actual AN/UYK-7 hardware and Helen Fletcher for assistance in the preparation of this document.

This report was prepared in the Programming Systems Branch of the Computer Programming Division and reviewed by Hermon W. Thombs, Head, Programming Systems Branch.

Released by:



R. T. RYLAND, JR., Head  
Strategic Systems Department

Accession for	
NTIS GRANT	
DDC TAB	
Unannounced	
Justification _____	
By _____	
Distribution _____	
Availability Codes _____	
Distr	Avail and/or special
A	

TABLE OF CONTENTS

AN/UYK-7 CP SEQUENCES	2
CP MAIN LOOP	3
I_SEQUENCE	4
INTERRUPT_SCAN	5
DECODE	6
DECODE_PHASE_TWO	7
INTERRUPT_SEQUENCE	8
REPEAT_SEQUENCE	9
CP SUPPORT ROUTINES LEVEL I	10
GENERATE_SYNCHRONOUS_INTERRUPT	11
GET_ISC	12
JUMP_ADDRESS	13
OP_READ	14
OP_STORE	15
CP SUPPORT ROUTINES LEVEL II	16
BPP_CHECK	17
IA_SEQUENCE	18
MEMORY_READ	19
SPR_CHECK	20
CP INSTRUCTION SUPPORT ROUTINES	21
CP/IOC_CLOCK_COMMUNICATIONS	22
DO_JUMP	23
GET_SHIFT_AMOUNT	24
HALF-WORD_TOGGLE	25
REPLACE_CHECK	26
_REPLACE	27
UPDATE_REPLACE	28
SET_CDL	29
SET_CDL2	30
CP UTILITY Routines	31
ADD_S	32
GET_ARES	33
GET_BRES	34
GET_SPECS	35
PUT_ARES	36
PUT_BRES	37
PUT_SPECS	38
FLOATING POINT SUBROUTINES	39
FLOATING_ADC_SUBTRACT_HEADER	40
FLOATING_OVERFLOW	41
FLOATING_NORMALIZE	42
FLOATING_ROUND	43
ROUND_UP	44
DIVIDE_COMPARE	45
FLOATING_POINT_END	46

CP	INSTRUCTION SET
-OR	
-SC	
-MS	
-MUR	
-ALP	
-LLP	
-CMT	
-TR	
-SLP	
-SSUM	
-SDIF	
-DS	
-TSF	
-DL	
DA	*
-DC	
-LBMP	
-FA	
-FAN	
-FH	
-FD	
-LX5	
-IPI	
-AEI	
-LIM	
-IO	
-IR	
-SP	
-LA	
-LXB	
-LDIF	
-AMA	
-AAA	
-LSUM	
-LNA	
-LM	
-LB	
-AB	
-ANG	
-SB	
-SA	
-SX8	
-SMA	
-SH	
-S2	
-RA	
-RI	
-W	*
-MULTPLY	
-DIVIDE	
-D	
-BC	
-CXI	
-C	

**AN/UYK-7 (CP)**  
**TABLE OF CONTENTS**

-CL	192
-CH	103
-CG	104
-JEP	105
-DJZ	106
-DJN2	107
-F91	108
-L8J	109
-JBN2	110
-JS	111
-JL	112
-JS3G	113
-HSC-60	119
-MLC-61	116
-MLC	115
-HDLC	116
-HSF	117
-MDSF	118
-MCP	119
-MDCP	120
-MOR	121
-MR	122
-HO	123
-MBT	124
-DL_SORT	125
-HLB	126
-HC	127
-MCL	128
-MCH	129
-HCB	130
-HSIR	131
-HSTC	132
-MPI	133
-H776	134
<b>AN/UYK-7 CP EMULATION GLOSSARY</b>	
<b>AN/UYK-7 CP EMULATION STATUS/CONTROL INDICATORS</b>	
<b>STATUS/CONTROL INDICATORS PAGE 2</b>	
<b>AN/UYK-7 CP ARCHITECTURE DESCRIPTORS</b>	
<b>AN/UYK-7 CP MAINTENANCE PANEL SUPPORT</b>	
<b>AN/UYK-7 CP EMULATION INTERNAL REGISTER SUPPORT</b>	
<b>DISTRIBUTION</b>	
<b>DISTRIBUTION LIST (GOVERNMENT INSTALLATIONS)</b>	
<b>DISTRIBUTION LIST (GOVERNMENT INSTALLATIONS CONTINUED)</b>	
<b>DISTRIBUTION LIST (PRIVATE INDUSTRY)</b>	
<b>DISTRIBUTION LIST (LOCAL)</b>	
<b>REFERENCES</b>	
<b>REFERENCES</b>	

N5MC AN/UYK-7 (CPI  
TABLE OF CONTENTS

	14 DEC 70	PAGE
DATA INDEX .....	.....	154
FLOW SEGMENT INDEX .....	.....	155

MSAC AN/UAK-7 (CP)

2 PAGE 30 DEC 14

AN/UAK-7 CP STANDARDS

## CP MAIN LOOP ..MAIN LOOP FOR AN/UYK-7 CENTRAL PROCESSOR

```

REF PAGE
14   * 1 DC FOREVER --LOOP THROUGH INSTRUCTION, INDIRECT ADDRESS, OPERAND & INTERRUPT SEQUENCES.
15   * 2 IF -STEP SET, THEN
16   * 3 SUSPEND LOOPING UNTIL RESTARTED
17   * 4 ENDIE
18   * 5 IF ASR(13,1) =CLASS IS LOGICOUT SET" .OR.
19   * 6 ASR(11,1) =INTERRUPT BASE REGISTERS SELECTED" .OR.
20   * 7 ASR(9,1) =MEMORY LOGICOUT INHIBIT SET", THEN
21   * 8 DISABLE SPR_CHECKS
22   * 9
23   * 10 ENABLE SPR_CHECKS
24   * 11
25   * 12 IF NOT REPEAT IN-PROGRESS, THEN
26   * 13 CLEAR MEMORY_STORE_INDICATOR -- THIS INDICATOR IS SET BY INSTRUCTIONS
27   * 14 ..THAT STORE INTO MEMORY. IT IS USED IN THE REPEAT TERMINATION LOGIC.
28   * 15 CALL I_SEQUENCE --FETCH INSTRUCTION, PERFORM BREAKPOINT & SPR CHECKS.
29   * 16
30   * 17 IF INTERRUPT_SCAN_INHIBIT CLEAR, THEN
31   * 18 CALL INTERRUPT_SCAN --CHECK FOR ASYNCHRONOUS INTERRUPTS.
32   * 19
33   * 20 CALL _DECODE ..EXECUTE INSTRUCTION INDICATED BY OPCODE FUNCTION DESIGNATORS.
34   * 21 IF REPEAT IN PROGRESS, THEN
35   * 22 CALL REPEAT_SEQUENCE --PERFORM REPEAT TERMINATION LOGIC.
36   * 23
37   * 24 IF ASR(15,1) =LOWER HALF-WORD INSTRUCTION", THEN
38   * 25 SET INTERRUPT_SCAN_INHIBIT ..DON'T ALLOW ASYNCHRONOUS INTERRUPTS
39   * 26 ..BETWEEN HALF-WORD INSTRUCTIONS.
40   * 27
41   * 28 ELSE
42   * 29 CLEAR INTERRUPT_SCAN_INHIBIT
43   * 30
44   * 31 ENDIE
45   * 32 IF CP MONITOR CLOCK POSITIVE (I.E. =+0), THEN
46   * 33 DECREMENT CP MONITOR CLOCK EACH 1/1024 SECOND
47   * 34 IF CP MONITOR CLOCK NEGATIVE, THEN
48   * 35 GENERATE CP MONITOR CLOCK INTERRUPT TO BE DETECTED BY INTERRUPT_SCAN
49   * 36
50   * 37 ENDIE

```

I\_SEQUENCE

REF	PAGE	CODE
56	1	** MOVE THE NEXT INSTRUCTION FROM MEMORY INTO THE U REGISTER.
57	2	** PERFORM APPROPRIATE BREAKPOINT AND SOFTWARE PROTECTION REGISTER (SPR) CHECKS.
59	3	32_REG(31) = P(0) - ASRC(15:1) ** THE PROPER DISPLACEMENT TO THE NEXT INSTRUCTION.
61	4	** REFETCH U ON A LOWER HALF-WORD SINCE A SYNCHRONOUS INTERRUPT MAY
62	5	** HAVE BEEN PROCESSED SINCE EXECUTION OF THE UPPER HALF-WORD INSTRUCTION.
64	6	32_REG(11) = 32_REG(3) ** GET TEMPORARY COPY OF DISPLACEMENT.
65	7	CALL ADD_S (P(S), 32_REG(11)) ** CONDITIONALLY ADD IN S REGISTER.
66	8	IF ASRC(15,1)=0 **NOT EXECUTING LOWER HALF OF INSTRUCTION*, THEN
67	9	CALL OPP_CHECK (32_REG(11), INSITUATION ADDRESS) **CHECK FOR BREAKPOINT.
68	10	ENDIF
69	11	CALL SPR_CHECK (P(S)) INSTRUCTION EXECUTE, "P=0", 32_REG(3))
70	12	CALL MEMORY READ (32_REG(11), U, P(S)) INSTRUCTION, "P=0")
71	13	IE ASRC(15,1) SET "EXECUTING LOWER HALF-WORD INSTRUCTION", THEN
72	14	UU := UL **SHIFT LOWER HALF-WORD INSTRUCTION
73	15	ENDIF
74	16	CLEAR EXECUTE_REMOTE_IN_PROGRESS
75	17	CLEAR CHARACTER_ADDRESSING_OVERFLOW
76	18	CLEAR SPR_PRIVILEGED_INSTRUCTION **ASSUME NO SPECIAL PRIVILEGED INSTRUCTION CHECKS
78	19	PCC1 = 32_REG(3)+1 **INCREMENT P TO NEXT INSTRUCTION ADDRESS.
79	20	DEFINE INSTRUCTION_FORMAT_INDICATOR **I.E. I, II, III, OR IV.
80	21	JE REPEAT_PENDING, THEN
81	22	IE INSTRUCTION_REPEATABLE (AS PER REPERTOIRE CARD), THEN
82	23	SET REPEAT_IN_PROGRESS
83	24	CLEAR REPEAT_PENDING
84	25	SET SPR_PRIVILEGED_INSTRUCTION **ALL REPEATED INSTRUCTIONS PRIVILEGED WHEN
86	26	** SPR(16:11) SET "INTERRUPT_BES FOR INDIRECT ADDRESSING" AND INDIRECT ADDRESSING.
88	27	IE EC>ULF>3 "CMP REFERENCE INSTRUCTION" **DON'T ALLOW DETECTION OF ASYNCHRONOUS INTERRUPTS
89	28	SET INTERRUPT_SCAN_INHIBIT **WHEN REPEATING A CMR REFERENCE INSTRUCTION.
91	29	ENDIF
92	30	ENDIF
93	31	ELSE
94	32	CLEAR REPEAT_PENDING
95	33	ABORT THE INSTRUCTION
96	34	ENDIF
97	35	RETURN
98	36	



## \_DECODE

```
* 1  ** DECODE INSTRUCTION IN U BASED 1 U(F). SOME INSTRUCTIONS REQUIRE
* 2  ** SUBFUNCTION FIELDS (SEE DECODE_HASE_TWO).
* 3  IF U(S)=7 .AND. ASR(B,1)=1 .AND. C.NCT. (OMP INSTRUCTION) .AND. TASK
* 4  MODE, THEN
* 5  CALL GENERATE_SYNCHRONOUS_INTERRUPT
* 6  (*P=1, PRIVILEGED INSTRUCTION, DUMPY_CODE) .ABORT INSTRUCTION.
* 7  ENDIF
* 8
* 9  DC CASE U(F) CORRESPONDING TO THE FOLLOWING TABLE
*10  ILLEGAL,   F01,    F02,  ILLEGAL,   F03,  F05,   F06,   F07,
*11  ILLEGAL,   F08,  ILLEGAL,   F09,  ILLEGAL,   F0A,  ILLEGAL,   F0B,
*12  ILLEGAL,   F0C,  ILLEGAL,   F0D,  ILLEGAL,   F0E,  ILLEGAL,   F0F,
*13  -LA,   -LXB,  -LDIF,  -AAA,  -AAA,  -LSUP,  -LNA,  -LR,
*14  -LB,   -LB,  -MM,  -SB,  -SA,  -SXBB,  -SNA,  -SM,
*15  ILLEGAL,  ILLEGAL,  -B2_BS,  -B2_BS,  -PA,  -PL,  -PAH,  -FC,
*16  ILLEGAL,  ILLEGAL,  -B2_BS,  -B2_BS,  -PA,  -PL,  -PAH,  -FC,
*17  ILLEGAL,  ILLEGAL,  -B2_BS,  -B2_BS,  -PA,  -PL,  -PAH,  -FC,
*18  -P,   -D,  -BC,  -CIL,  -C,  -CL,  -CP,  -CG,
*19  F5C,  -F51,  F52,  F53,  -LCT,  -LCI,  -SCI,  -SC1,
*20  HSC_BG,  HLC_B1,  -HLC,  -HLC,  -HBL,  -HPS,  -HDS,
*21
*22
*23
*24
*25
*26
*27
*28
*29  ILLEGAL,
*30  CALL GENERATE_SYNCHRONOUS_INTERRUPT
*31  (*P=1, CP ILLEGAL INSTRUCTION, DUMPY_CODE) .ABORT INSTRUCTION.
*32
```

DECODE-MHASE-TWO

## INTERRUPT\_SEQUENCE\_L\_CLASS,\_ISCI

REF

PAGE

```

210   * 1   ** P MODIFICATION PERFORMED PRIOR TO ENTRY.
211   * 2   . THERE IS NO PROVISION FOR POWER TOLERANCE INTERRUPTS (I.E.
212   * 3   . NEVER OCCUR, THUS NEVER HANDLED) IN THIS EMULATION.
213   * 4   . (1) - CLASS - THE CLASS (LEVEL) OF INTERRUPT BEING PROCESSED.
214   * 5   . (2) - ISCI - INTERRUPT STATUS CODE ASSOCIATED WITH THIS INTERRUPT.
215   * 6   ASR -> CPR (0*1350 + 4* _CLASS) ** SAVE ACTIVE STATUS REGISTEP.
216   * 7   ISC -> CMR (0*1360 + 4* _CLASS) ** SAVE INTERRUPT STATUS CODE.
217   * 8   P -> CMR (0*1370 + 4* _CLASS) **SAVE PROGRAM COUNTER.
218   * 9   DO CASE -CLASS OF
219   * 10  * 10  * MUST BE NON-OWNER TOLERANCE (SEE COMMENT ABOVE).
220   * 11  * 11  * ASR(19,203+A.0*402000) **LEAVE CP10, CLASS I,III,LOCKOUTS.
221   * 12  * 12  * SET INTERRUPT S&B SELECTS, MEMORY LOCKOUT INHIBIT, LBPP ENABLE AND BOOTSTRAP.
222   * 13  * 13  * PSJ := 7; P(D) := 0 **FORCE NDPO SWITCHER ACTIVATION.
223   * 14
224   * 15  \T1\ ASR(19,203+A.0*402000) **LEAVE CP10, CLASS I,LOCKOUTS,
225   * 16  *AND BUGGISTRAP BITS UNCHANGED.
226   * 17  ASR(19,203) = ASR(19,203).V.C*1374000' **SET STATE II,CLASS II,III,LOCKOUTS.
227   * 18  *SET INTERRUPT S&B SELECTS, MEMORY LOCKOUT INHIBIT & LBPP ENABLE.
228   * 19  *IF AUTO RECOVERY S<3 SELECTED .AND. _ISC-B'001C, "ILLEGAL INSTRUCTION", THEN
229   * 20  ASR(7,1) := 1 **SET HOBISTRAP MODE.
230   * 21  P(SJ) := 7; P(D) := 1 + 800STRAP_SWITCH SETTING ..II.E. 1,2,OR 3
231   ELSE
232     P:= CMR(0*1640) **CLASS II ICW.
233   ENDIF
234   * 22
235   * 23
236   * 24
237   * 25
238   * 26
239   * 27
240   * 28
241   * 29
242   * 30
243   * 31
244   * 32
245   * 33
246   * 34
247   * 35
248   * 36
249   * 37
250   * 38
251   * 39
252   * 40
253   * 41
254   * 42
255   * 43
256   * 44
257   * 45
258   * 46
259   * 47
260   * 48
261   * 49
262   * 50
263   * 51
264   * 52
265   * 53
266   * 54
267   * 55
268   * 56
269   * 57
270   * 58
271   * 59
272   * 60
273   * 61
274   * 62
275   * 63
276   * 64
277   * 65
278   * 66
279   * 67
280   * 68
281   * 69
282   * 70
283   * 71
284   * 72
285   * 73
286   * 74
287   * 75
288   * 76
289   * 77
290   * 78
291   * 79
292   * 80
293   * 81
294   * 82
295   * 83
296   * 84
297   * 85
298   * 86
299   * 87
300   * 88
301   * 89
302   * 90
303   * 91
304   * 92
305   * 93
306   * 94
307   * 95
308   * 96
309   * 97
310   * 98
311   * 99
312   * 100
313   * 101
314   * 102
315   * 103
316   * 104
317   * 105
318   * 106
319   * 107
320   * 108
321   * 109
322   * 110
323   * 111
324   * 112
325   * 113
326   * 114
327   * 115
328   * 116
329   * 117
330   * 118
331   * 119
332   * 120
333   * 121
334   * 122
335   * 123
336   * 124
337   * 125
338   * 126
339   * 127
340   * 128
341   * 129
342   * 130
343   * 131
344   * 132
345   * 133
346   * 134
347   * 135
348   * 136
349   * 137
350   * 138
351   * 139
352   * 140
353   * 141
354   * 142
355   * 143
356   * 144
357   * 145
358   * 146
359   * 147
360   * 148
361   * 149
362   * 150
363   * 151
364   * 152
365   * 153
366   * 154
367   * 155
368   * 156
369   * 157
370   * 158
371   * 159
372   * 160
373   * 161
374   * 162
375   * 163
376   * 164
377   * 165
378   * 166
379   * 167
380   * 168
381   * 169
382   * 170
383   * 171
384   * 172
385   * 173
386   * 174
387   * 175
388   * 176
389   * 177
390   * 178
391   * 179
392   * 180
393   * 181
394   * 182
395   * 183
396   * 184
397   * 185
398   * 186
399   * 187
400   * 188
401   * 189
402   * 190
403   * 191
404   * 192
405   * 193
406   * 194
407   * 195
408   * 196
409   * 197
410   * 198
411   * 199
412   * 200
413   * 201
414   * 202
415   * 203
416   * 204
417   * 205
418   * 206
419   * 207
420   * 208
421   * 209
422   * 210
423   * 211
424   * 212
425   * 213
426   * 214
427   * 215
428   * 216
429   * 217
430   * 218
431   * 219
432   * 220
433   * 221
434   * 222
435   * 223
436   * 224
437   * 225
438   * 226
439   * 227
440   * 228
441   * 229
442   * 230
443   * 231
444   * 232
445   * 233
446   * 234
447   * 235
448   * 236
449   * 237
450   * 238
451   * 239
452   * 240
453   * 241
454   * 242
455   * 243
456   * 244
457   * 245
458   * 246
459   * 247
460   * 248
461   * 249
462   * 250
463   * 251
464   * 252
465   * 253
466   * 254
467   * 255
468   * 256
469   * 257
470   * 258
471   * 259
472   * 260
473   * 261
474   * 262
475   * 263
476   * 264
477   * 265
478   * 266
479   * 267
480   * 268
481   * 269
482   * 270
483   * 271
484   * 272
485   * 273
486   * 274
487   * 275
488   * 276
489   * 277
490   * 278
491   * 279
492   * 280
493   * 281
494   * 282
495   * 283
496   * 284
497   * 285
498   * 286
499   * 287
500   * 288
501   * 289
502   * 290
503   * 291
504   * 292
505   * 293
506   * 294
507   * 295
508   * 296
509   * 297
510   * 298
511   * 299
512   * 300
513   * 301
514   * 302
515   * 303
516   * 304
517   * 305
518   * 306
519   * 307
520   * 308
521   * 309
522   * 310
523   * 311
524   * 312
525   * 313
526   * 314
527   * 315
528   * 316
529   * 317
530   * 318
531   * 319
532   * 320
533   * 321
534   * 322
535   * 323
536   * 324
537   * 325
538   * 326
539   * 327
540   * 328
541   * 329
542   * 330
543   * 331
544   * 332
545   * 333
546   * 334
547   * 335
548   * 336
549   * 337
550   * 338
551   * 339
552   * 340
553   * 341
554   * 342
555   * 343
556   * 344
557   * 345
558   * 346
559   * 347
560   * 348
561   * 349
562   * 350
563   * 351
564   * 352
565   * 353
566   * 354
567   * 355
568   * 356
569   * 357
570   * 358
571   * 359
572   * 360
573   * 361
574   * 362
575   * 363
576   * 364
577   * 365
578   * 366
579   * 367
580   * 368
581   * 369
582   * 370
583   * 371
584   * 372
585   * 373
586   * 374
587   * 375
588   * 376
589   * 377
590   * 378
591   * 379
592   * 380
593   * 381
594   * 382
595   * 383
596   * 384
597   * 385
598   * 386
599   * 387
599   * 388
599   * 389
599   * 390
599   * 391
599   * 392
599   * 393
599   * 394
599   * 395
599   * 396
599   * 397
599   * 398
599   * 399
599   * 400
599   * 401
599   * 402
599   * 403
599   * 404
599   * 405
599   * 406
599   * 407
599   * 408
599   * 409
599   * 410
599   * 411
599   * 412
599   * 413
599   * 414
599   * 415
599   * 416
599   * 417
599   * 418
599   * 419
599   * 420
599   * 421
599   * 422
599   * 423
599   * 424
599   * 425
599   * 426
599   * 427
599   * 428
599   * 429
599   * 430
599   * 431
599   * 432
599   * 433
599   * 434
599   * 435
599   * 436
599   * 437
599   * 438
599   * 439
599   * 440
599   * 441
599   * 442
599   * 443
599   * 444
599   * 445
599   * 446
599   * 447
599   * 448
599   * 449
599   * 450
599   * 451
599   * 452
599   * 453
599   * 454
599   * 455
599   * 456
599   * 457
599   * 458
599   * 459
599   * 460
599   * 461
599   * 462
599   * 463
599   * 464
599   * 465
599   * 466
599   * 467
599   * 468
599   * 469
599   * 470
599   * 471
599   * 472
599   * 473
599   * 474
599   * 475
599   * 476
599   * 477
599   * 478
599   * 479
599   * 480
599   * 481
599   * 482
599   * 483
599   * 484
599   * 485
599   * 486
599   * 487
599   * 488
599   * 489
599   * 490
599   * 491
599   * 492
599   * 493
599   * 494
599   * 495
599   * 496
599   * 497
599   * 498
599   * 499
599   * 500
599   * 501
599   * 502
599   * 503
599   * 504
599   * 505
599   * 506
599   * 507
599   * 508
599   * 509
599   * 510
599   * 511
599   * 512
599   * 513
599   * 514
599   * 515
599   * 516
599   * 517
599   * 518
599   * 519
599   * 520
599   * 521
599   * 522
599   * 523
599   * 524
599   * 525
599   * 526
599   * 527
599   * 528
599   * 529
599   * 530
599   * 531
599   * 532
599   * 533
599   * 534
599   * 535
599   * 536
599   * 537
599   * 538
599   * 539
599   * 540
599   * 541
599   * 542
599   * 543
599   * 544
599   * 545
599   * 546
599   * 547
599   * 548
599   * 549
599   * 550
599   * 551
599   * 552
599   * 553
599   * 554
599   * 555
599   * 556
599   * 557
599   * 558
599   * 559
599   * 560
599   * 561
599   * 562
599   * 563
599   * 564
599   * 565
599   * 566
599   * 567
599   * 568
599   * 569
599   * 570
599   * 571
599   * 572
599   * 573
599   * 574
599   * 575
599   * 576
599   * 577
599   * 578
599   * 579
599   * 580
599   * 581
599   * 582
599   * 583
599   * 584
599   * 585
599   * 586
599   * 587
599   * 588
599   * 589
599   * 590
599   * 591
599   * 592
599   * 593
599   * 594
599   * 595
599   * 596
599   * 597
599   * 598
599   * 599
599   * 600
599   * 601
599   * 602
599   * 603
599   * 604
599   * 605
599   * 606
599   * 607
599   * 608
599   * 609
599   * 610
599   * 611
599   * 612
599   * 613
599   * 614
599   * 615
599   * 616
599   * 617
599   * 618
599   * 619
599   * 620
599   * 621
599   * 622
599   * 623
599   * 624
599   * 625
599   * 626
599   * 627
599   * 628
599   * 629
599   * 630
599   * 631
599   * 632
599   * 633
599   * 634
599   * 635
599   * 636
599   * 637
599   * 638
599   * 639
599   * 640
599   * 641
599   * 642
599   * 643
599   * 644
599   * 645
599   * 646
599   * 647
599   * 648
599   * 649
599   * 650
599   * 651
599   * 652
599   * 653
599   * 654
599   * 655
599   * 656
599   * 657
599   * 658
599   * 659
599   * 660
599   * 661
599   * 662
599   * 663
599   * 664
599   * 665
599   * 666
599   * 667
599   * 668
599   * 669
599   * 670
599   * 671
599   * 672
599   * 673
599   * 674
599   * 675
599   * 676
599   * 677
599   * 678
599   * 679
599   * 680
599   * 681
599   * 682
599   * 683
599   * 684
599   * 685
599   * 686
599   * 687
599   * 688
599   * 689
599   * 690
599   * 691
599   * 692
599   * 693
599   * 694
599   * 695
599   * 696
599   * 697
599   * 698
599   * 699
599   * 700
599   * 701
599   * 702
599   * 703
599   * 704
599   * 705
599   * 706
599   * 707
599   * 708
599   * 709
599   * 710
599   * 711
599   * 712
599   * 713
599   * 714
599   * 715
599   * 716
599   * 717
599   * 718
599   * 719
599   * 720
599   * 721
599   * 722
599   * 723
599   * 724
599   * 725
599   * 726
599   * 727
599   * 728
599   * 729
599   * 730
599   * 731
599   * 732
599   * 733
599   * 734
599   * 735
599   * 736
599   * 737
599   * 738
599   * 739
599   * 740
599   * 741
599   * 742
599   * 743
599   * 744
599   * 745
599   * 746
599   * 747
599   * 748
599   * 749
599   * 750
599   * 751
599   * 752
599   * 753
599   * 754
599   * 755
599   * 756
599   * 757
599   * 758
599   * 759
599   * 760
599   * 761
599   * 762
599   * 763
599   * 764
599   * 765
599   * 766
599   * 767
599   * 768
599   * 769
599   * 770
599   * 771
599   * 772
599   * 773
599   * 774
599   * 775
599   * 776
599   * 777
599   * 778
599   * 779
599   * 780
599   * 781
599   * 782
599   * 783
599   * 784
599   * 785
599   * 786
599   * 787
599   * 788
599   * 789
599   * 790
599   * 791
599   * 792
599   * 793
599   * 794
599   * 795
599   * 796
599   * 797
599   * 798
599   * 799
599   * 800
599   * 801
599   * 802
599   * 803
599   * 804
599   * 805
599   * 806
599   * 807
599   * 808
599   * 809
599   * 810
599   * 811
599   * 812
599   * 813
599   * 814
599   * 815
599   * 816
599   * 817
599   * 818
599   * 819
599   * 820
599   * 821
599   * 822
599   * 823
599   * 824
599   * 825
599   * 826
599   * 827
599   * 828
599   * 829
599   * 830
599   * 831
599   * 832
599   * 833
599   * 834
599   * 835
599   * 836
599   * 837
599   * 838
599   * 839
599   * 840
599   * 841
599   * 842
599   * 843
599   * 844
599   * 845
599   * 846
599   * 847
599   * 848
599   * 849
599   * 850
599   * 851
599   * 852
599   * 853
599   * 854
599   * 855
599   * 856
599   * 857
599   * 858
599   * 859
599   * 860
599   * 861
599   * 862
599   * 863
599   * 864
599   * 865
599   * 866
599   * 867
599   * 868
599   * 869
599   * 870
599   * 871
599   * 872
599   * 873
599   * 874
599   * 875
599   * 876
599   * 877
599   * 878
599   * 879
599   * 880
599   * 881
599   * 882
599   * 883
599   * 884
599   * 885
599   * 886
599   * 887
599   * 888
599   * 889
599   * 890
599   * 891
599   * 892
599   * 893
599   * 894
599   * 895
599   * 896
599   * 897
599   * 898
599   * 899
599   * 900
599   * 901
599   * 902
599   * 903
599   * 904
599   * 905
599   * 906
599   * 907
599   * 908
599   * 909
599   * 910
599   * 911
599   * 912
599   * 913
599   * 914
599   * 915
599   * 916
599   * 917
599   * 918
599   * 919
599   * 920
599   * 921
599   * 922
599   * 923
599   * 924
599   * 925
599   * 926
599   * 927
599   * 928
599   * 929
599   * 930
599   * 931
599   * 932
599   * 933
599   * 934
599   * 935
599   * 936
599   * 937
599   * 938
599   * 939
599   * 940
599   * 941
599   * 942
599   * 943
599   * 944
599   * 945
599   * 946
599   * 947
599   * 948
599   * 949
599   * 950
599   * 951
599   * 952
599   * 953
599   * 954
599   * 955
599   * 956
599   * 957
599   * 958
599   * 959
599   * 960
599   * 961
599   * 962
599   * 963
599   * 964
599   * 965
599   * 966
599   * 967
599   * 968
599   * 969
599   * 970
599   * 971
599   * 972
599   * 973
599   * 974
599   * 975
599   * 976
599   * 977
599   * 978
599   * 979
599   * 980
599   * 981
599   * 982
599   * 983
599   * 984
599   * 985
599   * 986
599   * 987
599   * 988
599   * 989
599   * 990
599   * 991
599   * 992
599   * 993
599   * 994
599   * 995
599   * 996
599   * 997
599   * 998
599   * 999
599   * 1000
599   * 1001
599   * 1002
599   * 1003
599   * 1004
599   * 1005
599   * 1006
599   * 1007
599   * 1008
599   * 1009
599   * 1010
599   * 1011
599   * 1012
599   * 1013
599   * 1014
599   * 1015
599   * 1016
599   * 1017
599   * 1018
599   * 1019
599   * 1020
599   * 1021
599   * 1022
599   * 1023
599   * 1024
599   * 1025
599   * 1026
599   * 1027
599   * 1028
599   * 1029
599   * 1030
599   * 1031
599   * 1032
599   * 1033
599   * 1034
599   * 1035
599   * 1036
599   * 1037
599   * 1038
599   * 1039
599   * 1040
599   * 1041
599   * 1042
599   * 1043
599   * 1044
599   * 1045
599   * 1046
599   * 1047
599   * 1048
599   * 1049
599   * 1050
599   * 1051
599   * 1052
599   * 1053
599   * 1054
599   * 1055
599   * 1056
599   * 1057
599   * 1058
599   * 1059
599   * 1060
599   * 1061
599   * 1062
599   * 1063
599   * 1064
599   * 1065
599   * 1066
599   * 1067
599   * 1068
599   * 1069
599   * 1070
599   * 1071
599   * 1072
599   * 1073
599   * 1074
599   * 1075
599   * 1076
599   * 1077
599   * 1078
599   * 1079
599   * 1080
599   * 1081
599   * 1082
599   * 1083
599   * 1084
599   * 1085
599   * 1086
599   * 1087
599   * 1088
599   * 1089
599   * 1090
599   * 1091
599   * 1092
599   * 1093
599   * 1094
599   * 1095
599   * 1096
599   * 1097
599   * 1098
599   * 1099
599   * 1100
599   * 1101
599   * 1102
599   * 1103
599   * 1104
599   * 1105
599   * 1106
599   * 1107
599   * 1108
599   *
```

AN/UVK-7 (CP)  
AN/UTE-7 CP SEQUENCES

## REPEAT\_SEQUENCE

```

REF PAGE
*   1
*   2   ** PERFORM TERMINATION LOGIC.
*   3   B17) = B17) - 1. DECREMENT REPEAT_COUNTER.
*   4   BIU(B1) = BIU(B1) + REPEAT_SV **INCREMENT B REGISTER WITH SY FIELD OF REPEAT_INSTRUCTION
*   5   IF B17 = 0, THEN
*       CLEAR REPEAT_IN_PROGRESS INDICATION
*   6   ELSE **CHECK REPEAT TERMINATION CONDITIONS (OTHER THAN B17=0).
*       IF (U(31,6)>42 .AND. U(31,6)<90) .OR. (U(31,6)=03 .AND. U(22,3)=7) "TSF", THEN --A
*           COMPARE INSTRUCTION.
*           DQ CASE REPEAT_A(B4) OF
*               V0 IF ASR(2,1)=0 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V1 IF ASR(2,1)=1 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V2 IF ASR(2,2)=0'01' "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V3 IF ASR(1,1)=1 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V4 IF ASR(1,1)=0 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V5 IF ASR(2,1)=0 .OR. ASR(1,1)=0 "<>", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V6 IF ASR(2,1)=1 "OUT OF LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V7 IF ASR(0,1)=0 "WITHIN LIMITS", THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*           ENDQ **NOT A COMPARE INSTRUCTION.
*           DQ CASE REPEAT(B4) OF
*               V0 IF REPEAT_ACCUMULATOR >>0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V1 IF REPEAT_ACCUMULATOR =0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V2 IF REPEAT_ACCUMULATOR >>0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V3 IF REPEAT_ACCUMULATOR <<0, THEN CLEAR REPEAT_IN_PROGRESS ENDIF
*               V4 CASE 4 IMPLIES DO NOT TERMINATE
*               V5 IF REPEAT_ACCUMULATOR EVEN PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR
*                   REPEAT_IN_PROGRESS ENDIF
*               V6 IF REPEAT_ACCUMULATOR ODD PARITY .AND. MEMORY_STORE_INDICATOR, THEN CLEAR
*                   REPEAT_IN_PROGRESS ENDIF
*               V7 CASE 7 IMPLIES DO NOT TERMINATE
*           ENDQ **END CASE.
*           ENDIE
*           IE = MCT.REPEAT_IN_PROGRESS, THEN
*               CLEAR_INTERRUPT_SCAN_INHIBIT_INDICATOR
*           ENDIE
*           RETURN
*   263   * 1
*   264   * 2
*   265   * 3
*   266   * 4
*   267   * 5
*   268   * 6
*   269   * 7
*   270   * 8
*   271   * 9
*   272   * 10
*   273   * 11
*   274   * 12
*   275   * 13
*   276   * 14
*   277   * 15
*   278   * 16
*   279   * 17
*   280   * 18
*   281   * 19
*   282   * 20
*   283   * 21
*   284   * 22
*   285   * 23
*   286   * 24
*   287   * 25
*   288   * 26
*   289   * 27
*   290   * 28
*   291   * 29
*   292   * 30
*   293   * 31
*   294   * 32
*   295   * 33
*   296   * 34

```

MSMC

AN/UYK-7 (CP)

14 DEC 79 PAGE 10

\* CP SUPPORT ROUTINES LEVEL 1 \*

.....

GENERATE\_SYNCHRONOUS\_INTERRUPT (P\_PMODIFICATION, \_INTERRUPT, \_CODE)

REF

```

PAGE
*   1   ** GENERATE THE APPROPRIATE SYNCHRONOUS INTERRUPT, BUILD THE INTERRUPT STATUS CODE (ISC).
308   *   2   ..AND MODIFY PROGRAM COUNTER FOR STORAGE AS INDICATED.
309   *   3   ..CURRENT INSTRUCTION IS ABORTED WHETHER OR NOT THE INTERRUPT IS ACTUALLY GENERATED.
311   *   4   ..(1) P_PMODIFICATION - P REGISTER MODIFICATION VALUE.
312   *   5   ..(2) -INTERRUPT - INTERRUPT INFORMATION.
313   *   6   ..(3) -CODE - INFORMATION FOR ISC GENERATION (MEMORY BANK, IDC#, ETC.).
```

315 \* 7 IF -INTERRUPT IS A CLASS II, THEN

316 \* 8 ..CLASS "I" -CLASS INDICATOR FOR INTERRUPT SEQUENCE.

317 \* 9 ..IF ASR(4,1) =CLASS "I" LOCKED OUT", THEN

318 \* 10 ..RETURN ..TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).

319 \* 11 ENDIF

320 \* 12 ELSE

321 \* 13 ..IE -INTERRUPT IS A CLASS III, THEN

322 \* 14 ..CLASS "III" -CLASS INDICATOR FOR INTERRUPT SEQUENCE.

323 \* 15 ..IF ASR(3,1) =CLASS III LOCKED OUT", THEN

324 \* 16 ..RETURN ..TO MAIN LOOP (LOCKED OUT IMPLIES NO INTERRUPT GENERATED).

325 \* 17 ENDIF

326 \* 18 ELSE ..MUST BE CLASS IV IN ISRT SINCE THERE ARE NO CLASS III SYNCHRONOUS INTERRUPTS.

327 \* 19 CLASS "IV" -CLASS INDICATOR FOR INTERRUPT SEQUENCE.

328 \* 20 ..NOTE: THE CLASS IV INTERRUPT (REI) IS NEVER LOCKED OUT.

329 \* 21 ENDIF

330 \* 22 ENDIF

331 \* 23 DO CASE -INTERRUPT OF

332 \* 24 ..OP MEMORY RESUME, ISC = B'00000 ..CODE = MEMORY BANK NUMBER (0-15).

333 \* 25 ..VOC COMMAND RESUME, ISC = B'00000001 ..CODE = IOC NUMBER.

334 \* 26 ..INSTRUCTION MEMORY RESUME, ISC = B'00100 ..CODE = MEMORY BANK NUMBER (0-15).

335 \* 27 ..VOC INTERRUPT CODE RESUME, ISC = B'00010 ..CODE = IOC NUMBER.

336 \* 28 ..FLOATING POINT ERROR, ISC = B'00001 ..CODE = IOC NUMBER.

337 \* 29 ..CP ILLEGAL INSTRUCTION ERROR, ISC = B'00001

338 \* 30 ..PRIVILEGED INSTRUCTION ERROR, ISC = B'00011

339 \* 31 ..OPERAND READ OR INDIRECT ADDRESSING ISC = B'00110

340 \* 32 ..OPERAND WRITE, ISC = B'10011

341 \* 33 ..OPERAND LIMIT, ISC = B'10101

342 \* 34 ..INSTRUCTION BREAKPOINT MATCH, ISC = B'10111

343 \* 35 ..INSTRUCTION EXECUTE, ISC = B'11011

344 \* 36 ..INSTRUCTION LIMIT, ISC = B'11110

345 \* 37 ..REI, ISC = ..CODE EQUIVALENT TO ISC FOR REI.

346 \* 38 ENDIF ..END CASE.

347 \* 39 DO CASE ..MODIFY PROGRAM COUNTER FOR INTERRUPT STORAGE.

348 \* 40 ..REPEAT\_PENDING, P = P-1 ..REEXECUTE REPEAT.

349 \* 41 ..REPEAT\_IN\_PROGRESS, P = P-2 ..REEXECUTE REPEAT.

350 \* 42 ELSE

351 \* 43 ..P = P-P\_MODIFICATION ..SO PROPER RETURN CAN BE MADE.

352 \* 44 ENDDO ..END CASE.

353 \* 45 CALL INTERRUPT\_SEQUENCE (CLASS, ISC)

354 \* 46 RETURN

355 \* 47

AN/UVK-7 (CP)  
CP SUPPORT ROUTINES LEVEL 1

## GET\_ISC\_I\_CLASS, REF \_CODEI

REF

PAGE

```

*   1   ** ACQUIRE INTERRUPT STATUS CODE (ISC) FROM APPROPRIATE IOC AND ADD
*   2   ** IOC NUMBER IN BITS 9 & 8 BEFORE RETURNING (ISC) TO CALLER. SHOULD
*   3   ** THE IOC RETURN AN INVALID ISC, THIS FACT MUST BE CONVEYED TO THE CALLER.
*   4   ** (1) - CLASS - INTERRUPT LEVEL (1 OR 3) OF CONCERN.
*   5   ** (2) - CODE - ISC RECEIVED FROM IOC AND RETURNED TO CALLER.
*   6   DETERMINE IOC # CAUSING INTERRUPT
*   7   RECOGNIZE RECEIPT OF INTERRUPT BY MAKING AN IOC REQUEST ON IOC_#8
*   8   SEND INTERRUPT CLASS DESIGNATOR (CLASS) VIA Q_BUS
*   9   WAIT UNTIL IOC ACQUIRES CLASS INDICATOR
* 10  RECEIVE ISC FROM IOC_# ACROSS Q_BUS
* 11  REACTIVATE CLASS(I-CLASS) INTERRUPT LINE
* 12  CODE := IOC_# .CON. ISC .>2-BIT IOC # CONCATENATED WITH 8-BIT ISC (POSSIBLY INVALID).
* 13  RETURN

```

NSWC ANS/TK-7 (CP)  
CP SUPPORT ROUTINES LEVEL 1

14 DEC 79 PAGE 13

JUMP\_ADDRESS(M\_DISPLACEMENT, REF\_OPERAND\_S, REF\_OPERAND\_DISPLACEMENT)

REF  
PAGE

```
378   * 1   ** CALLED BY JUMP TYPE INSTRUCTIONS. FINDS THE "JUMP TO" ADDRESS AND
379   * 2   ** PERFORMS BREAKPOINT AND SPR CHECKS.
380   * 3   ** (1) M_DISPLACEMENT - 0/1 FOR OP1/OP2 SEQUENCE RESPECTIVELY.
381   * 4   ** (2) OPERAND_S - THE BASE REGISTER PORTION OF THE ADDRESS.
382   * 5   ** (3) OPERAND_DISPLACEMENT - THE 16-BIT DISPLACEMENT PORTION OF THE ADDRESS.
383   * 6   ** IF UC(1) "INDIRECT ADDRESSING", THEN .DO CASCADING.
384   * 7   ** CALL IA-SEQUENCE (OPERAND_S, OPERAND_DISPLACEMENT, DUM_C, DUM_C1, DUM_P, DUM_MASK)
385   * 8   ** .DUM_C, P, MASK SINCE CHARACTER ADDRESSING HAS NO MEANING FOR JUMP INSTRUCTIONS.
386   * 9   ** OPERAND_DISPLACEMENT == OPERAND_DISPLACEMENT + M_DISPLACEMENT
387   * 10  *
388   * 11  OPERAND_DISPLACEMENT == UC(Y) + BLU(B)(15) + M_DISPLACEMENT
389   * 12  OPERAND_S == UC(S)
390   * 13  ENDIF
391   * 14  32_REG(1) = OPERAND_DISPLACEMENT // GET COPY FOR USE IN CHECKING.
392   * 15  CALL SPR_CHECK (OPERAND_S, INSTRUCTION_EXECUTE, "P-"), OPERAND_DISPLACEMENT // ECP(66).
393   * 16  CALL ADD_S (OPERAND_S, 32_REG(1)) // COMPUTE FINAL (ABSOLUTE) ADDRESS.
394   * 17  CALL MEMORY_READ (32_REG(1), DUMMY, OPERAND_S, "P-") // CHECK FOR OPERAND MEMORY RESUME.
395   * 18  CALL SPR_CHECK (32_REG(1), OPERAND) // CHECK FOR OPERAND BREAKPOINT.
396   * 19  RETURN
397   * 20  *
398   * 21  *
399   * 22  *
400   * 23  *
401   * 24  *
```

## OP\_READ (REF 32\_REG, M\_DISPLACEMENT)

```

REF PAGE *****
403   * 1  ** DETERMINE OPERAND ADDRESS FROM U REGISTER USING I, K, S, Y & INDICATORS.
405   * 2  ** TRANSFER THE 32-BIT OPERAND (AS DETERMINED BY OPERAND TYPE) TO THE REGISTER SUPPLIED BY CALLER.
407   * 3  ** 32_REG = 32-BIT REG - 32-BIT SEQUENCE.
408   * 4  ** (1) M_DISPLACEMENT - O/P FOR OP1/OP2 SEQUENCE RESPECTIVELY.
409   * 5  ** IF U(I) = INDIRECT ADDRESSING, THEN
410     * 6  CALL IA_SEQUENCE DESIGNATOR, Y, C1, C2, P, _MASK) **FIND FINAL OPERAND ADDRESS VALUES.
412   * 7  OPERAND_DISPLACEMENT = OPERAND_DISPLACEMENT + M_DISPLACEMENT
413   * 8  ELSE **COMPUTE FINAL OPERAND ADDRESS VALUES.
414   * 9    Y = U(Y)+M_DISPLACEMENT(K5) **ADD DISPLACEMENT AND INDEX REGISTER.
415   * 10   Y = Y+M_DISPLACEMENT **ADD IN M_DISPLACEMENT FOR DESIRED OP1 OR OP2 SEQUENCE.
417   * 11   S_DESIGNATOR = U(S)
418   * 12   IF CHARACTER_ADDRESSING OVERRIDE, THEN
419     * 13     ACQUIRE_ACTIVE_C, _P, AND _MASK
420   * 14
421   * 15   ELSE C := 8*10 **FORCE FOLLOWING CODE TO USE K DESIGNATOR.
422   * 16
423   * 17 ENDIF
424   * 18 IF _NOT_ (INSTRUCTION_FORMAT_INDICATOR_I = AND_K=0 .AND. _C EVEN) "NOT IMMEDIATE OPERAND", THEN
425     * 19   CALL SPR_CHECK (S_DESIGNATOR, OPERAND READ, "P="1)
426     * 20   CALL ADD_S_IS_DESIGNATOR, Y1
427     * 21   CALL BPF_CHECK (Y, OPERAND)
428     * 22   CALL MEMORY_READ (Y, 32_REG, S_DESIGNATOR, OPERAND, "P="1)
429     * 23   ELSE "HANDLE IMMEDIATE OPERAND"
430     * 24     32_REG := U(S)
431     * 25   IF C > 0 .OR. _C1 > 0, THEN **IMMEDIATE OPERAND FCR IWS INDIRECT ADDRESS DOESN'T USE
432     * 26     BIU(L0).
433     * 27   32_REG := 32_REG + BIU(B0))
434   * 28 ENDIF
435   * 29
436   * 30 IF _C EVEN, THEN **NOT CHARACTER ADDRESSING.
437   * 31   IE INSTRUCTION_FORMAT_INDICATOR_I, THEN
438     * 32   ADJUST 32_REG ACCORDING TO U(K) ** INCLUDES IMMEDIATE OPERANDS.
439   * 33
440   * 34 ELSE **CHARACTER ADDRESSING.
441     * 35     RIGHT JUSTIFY / ZERO FILL 32_PEG.
442     * 36     .THIS CAN BE DONE BY SHIFTING 32_REG RIGHT LOGICAL_P BITS & ANDING WITH _MASK.
443   * 37 ENDIF
444   * 38 RETURN

```

MS/C AN/UYK-7 (CP)  
CP SUPPORT ROUTINES LEVEL I  
14 DEC 79 PAGE 15

#### OP\_STORE (32\_REG, #\_DISPLACEMENT)

```

REF PAGE *****
448   * 1   ** DETERMINE OPERAND ADDRESS FROM THE U REGISTER USING I, K, S, Y & # INDICATORS.
449   * 2   ** STORE THE QUANTITY SUPPLIED BY CALLER IN MEMORY.
450   * 3   ** (1) 32_REG - 32-BIT VALUE TO TRANSFER INTO MEMORY SUBJECT TO OPERAND TYPE RESTRICTIONS.
451   * 4   ** (2) #_DISPLACEMENT - #SI FOR OP1/OP2 SEQUENCE RESPECTIVELY.
452   * 5   ** IF US(I) = "INDIRECT ADDRESSING", THEN
453   * 6   **   CALL IA_SEQUENCE(S, DESIGNATOR, Y, _C, _CL, _P, _MASK) --FIND FINAL OPERAND ADDRESS VALUES.
454   * 7   ** OPERAND_DISPLACEMENT == OPERAND_DISPLACEMENT + #_DISPLACEMENT
455   * 8   ** ELSE COMPUTE FINAL OPERAND ADDRESS VALUES.
456   * 9   ** Y == U(Y)+#U(B)(Y)(15) --ADD DISPLACEMENT AND INDEX REGISTER.
457   * 10  ** Y == Y+#_DISPLACEMENT --ADD IN #_DISPLACEMENT FOR DESIRED OP1 OR OP2 SEQUENCE.
458   * 11  ** S, DESIGNATOR == US(S)
459   * 12  ** IF CHARACTER_ADDRESSING_OVERFLOW, THEN
460   * 13  **   ACQUIRE_ACTIVE _C, _P AND _MASK.
461   * 14  ** ELSE
462   * 15  **   C == 8'10' --FORCE FOLLOWING CODE TO USE K DESIGNATOR.
463   * 16  ** ENDFE
464   * 17  ** ENDOIF
465   * 18  ** SET MEMORY_STORE_INDICATOR --FOR USE IN REPEAT TERMINATION DETECTION.
466   * 19  ** IF NOT WORD_REFERENCE, THEN
467   * 20  **   IF NOT_INSTRUCTION_FORMAT_INDICATOR=1 .AND. K=0 .AND. C EVEN) "NOT IMMEDIATE OPERAND",
468   * 21  **   THEN
469   * 22  **     CALL SPA_CHECK_IS_DESIGNATOR, OPERAND_WRITE, "P-", Y, Y1
470   * 23  **     CALL ADD_S_IS_DESIGNATOR, Y;
471   * 24  **     CALL DPR_CHECK (Y, OPERAND)
472   * 25  **     CALL MEMORY_READ (Y, VALUE, S, DESIGNATOR, OPERAND, "P-", Y1)
473   * 26  **     IF _C EVEN, THEN --NOT CHARACTER ADDRESSING
474   * 27  **       IF INSTRUCTION_FORMAT_INDICATOR = 1, THEN
475   * 28  **         ADJUST 32_REG ACCORDING TO U(K) AND MASK INTO VALUE
476   * 29  **       ENDIF
477   * 30  **       *CHARACTER ADDRESSING
478   * 31  **       POSITION CHARACTER AND MASK INTO VALUE
479   * 32  **     ENDFE
480   * 33  **     TRANSFER VALUE TO MEMORY(Y1) --STORE TO WORD LOCATION IS A NOOP.
481   * 34  **     ENDFE --FORMAT 1 INSTRUCTION WITH K=0 IS A NOOP
482   * 35  ** RETURN

```

NSWC

ANNUAL-7 (CP)

14 DEC 79 PAGE 16

\* CP SUPPORT ROUTINES LEVEL III

## BPR\_CHECK (A000\_A000, TYPE\_)

REF PAGE .....

```

*   1   * 1. PERFORM BREAKPOINT REGISTER CHECKS. THE UYR7 EMULATION HAS EIGHT
* 90   * 2   * BREAKPOINT REGISTERS. ONE REGISTER CORRESPONDS TO THE ACTUAL HARDWARE
* 91   * 3   * BREAKPOINT REGISTER OF THE REAL UYR-7 MACHINE. THE OTHER SEVEN
* 92   * 4   * REGISTERS ARE PSEUDO BREAKPOINT REGISTERS AND ARE CONSIDERED
* 93   * 5   * EXTENSIONS OF THE UYR-7 ARCHITECTURE. THE PSEUDO BREAKPOINT REGISTERS
* 94   * 6   * ARE LOCATED IN CPP LOCATIONS 61-67. PSEUDO BREAKPOINTS ARE MOST
* 95   * 7   * ACCESSIBLE TO UYR-7 PROGRAMS). IF PSEUDO BREAKPOINTS ARE ENABLED
* 96   * 8   * ALL BREAKPOINT REGISTERS WILL BE CHECKED TOTAL OF 8!, OTHERWISE, ONLY
* 97   * 9   * THE EMULATED HARDWARE REGISTER WILL BE CHECKED. TYPE_ WILL BE TESTED
* 98   * 10  * AGAINST THE TYPE FIELD OF THE ACTIVE BREAKPOINT REGISTERS AND IF THEY
* 99   * 11  * MATCH, ABS_CADM WILL BE COMPARED TO THE BREAKPOINT REGISTER COMPARISON
* 100  * 12  * COEFFICIENTS 0-12!. FOR THE EMULATED HARDWARE REGISTER, A 'INIT' ON
* 101  * 13  * ACCESS COMPARISON CAUSES A 'HALT' OR CLASS II INTERRUPTION DEPENDING ON
* 102  * 14  * THE PREGAM/MANUAL SWITCH. FOR THE PSEUDO BREAKPOINT REGISTERS, A
* 103  * 15  * 'INIT' ON ADDRESS COMPARISON CAUSES A 'HALT'. REGARDLESS OF THE
* 104  * 16  * PREGAM/MANUAL SWITCH SETTING.
* 105  * 17  * ((1) ABS_A20 - 10 BIT ABSOLUTE UYR-7 ADDRESS
* 106  * 18  * (2) TYPE_ - TYPE OF BREAKPOINT CHECK DESIRED
* 107  * 19  * IE PSEUDO_BREAKPOINT IS ENABLED, THEN
* 108  * 20  * CNT = TOTAL # OF PSEUDO BREAKPOINTS - CHECK ALL BREAKPOINT REGISTERS.
* 109  * 21  * ELSE
* 110  * 22  * CNT = C ..ONLY EMULATED HARDWARE BREAKPOINT CHECKED.
* 111  * 23  * ENDIF
* 112  * 24  * DO WHILE CNT > 0 ..CYCLE THRU ACTIVE BREAKPOINT REGISTERS.
* 113  * 25  * IF C#BREAKPOINT_REGISTER+CNT<19,2> 0, THEN
* 114  * 26  *   IF (TYPE_==CUR_BREAKPOINT_REGISTER+CNT)<19,2> -OR-
* 115  * 27  *     CUR_BREAKPOINT_REGISTER+CNT<19,2>-31 "MATCHING TYPES".
* 116  * 28  *     IF ABS_A20 == CUR_BREAKPOINT_REGISTER+CNT<19,2> "MATCHING ADDRESS". THEN
* 117  * 29  *       IE MANUAL MODE -OR- CNT > 0 "PSEUDO BPT".
* 118  * 30  *       SET STEP =STOP IN MAIN LOOP IF OPERAND BREAKPOINT.
* 119  * 31  *     IE TYPE_ = INSTRUCTION BREAKPOINT, THEN
* 120  * 32  *       INSTRUCTION EXECUTION..STOP IMMEDIATELY IF INSTRUCTION TYPE_.
* 121  * 33  *     ELSE
* 122  * 34  *       IF ASR(13,1) "CLASS II LOCKED OUT", THEN
* 123  * 35  *         RETURN ..BREAKPOINT INTERRUPTS DISABLED.
* 124  * 36  *       ENDIF
* 125  * 37  *     ELSE
* 126  * 38  *       IF TYPE_ = INSTRUCTION BREAKPOINT, THEN
* 127  * 39  *         GENERATE SYNCHRONOUS_INTERRUPT (P=0, INSTRUCTION BREAKPOINT)
* 128  * 40  *         ..ABORT INSTRUCTION.
* 129  * 41  *       ELSE
* 130  * 42  *         GENERATE SYNCHRONOUS_INTERRUPT (P=1, OPERAND BREAKPOINT)
* 131  * 43  *         ..ABORT INSTRUCTION.
* 132  * 44  *       ENDIF
* 133  * 45  *     ENDIF
* 134  * 46  *   ENDIF
* 135  * 47  * ENDIF
* 136  * 48  * ENDIF
* 137  * 49  * ENDIF
* 138  * 50  * CNT = CNT-1
* 139  * 51  * ENDDO ..END WHILE.
* 140  * 52  * RETURN

```

IA-SEQUENCE (REF\_S\_DESIGNATOR,REF\_Y,REF\_C,REF\_OPERAND,REF\_MASK)

```

REF PAGE
543   * 1   ** PERFORMS INDIRECT ADDRESSING RETURNING THE FOLLOWING VALUES:
544   * 2   ** (1) S_DESIGNATOR - S DESIGNATOR OF OPERAND.
545   * 3   ** (2) Y - DISPLACEMENT POSITION OF THE OPERAND ADDRESS.
546   * 4   ** (3) C - P/E MASK - CHARACTER ADDRESSING PARAMETERS WHEN NEEDED.
547   * 5   ** CALLER IS RESPONSIBLE FOR ENSURING THAT AN IA-SEQUENCE IS REQUIRED.
548   * 6   ** Y = L(Y) + B(L(Y))15)
549   * 7   ** S_DESIGNATOR = U(S)
550   * 8   DO WHILE L(Y) "INDIRECTION" --CASCADE IGNORING U(Y).
551   * 9   CALL SPC_CHECK(S_DESIGNATOR,INDIRECT_ADDRESSING,-P-1,Y)
552   * 10  32_REG11 = Y --SAVE Y FOR POSSIBLE COMPARISON TO SPB.
553   * 11  CALL ADD_S(15,SPB,32_EG11)
554   * 12  CALL BPR_CHECK(32_REG11),OPERAND)
555   * 13  CALL MEMORY_READ32_PEC11,IACM,S_DESIGNATOR,OPERAND,"P-11" ..GET INDIRECT ADDRESS CONT'D
      LDO_IACM.
556   * 14  U(19,20) = IACM19,20) ..UPDATE THE U REGISTER B, I, S & Y FIELDS.
557   * 15  ** EXPERIMENTS INDICATE THAT U(19,17) IS ALWAYS SET TO 0 IF THE LAST IACM IS OF
558   * 16  ** THE SPECIAL BASE (16SI) TYPE. SINCE NO DOCUMENTATION CAN BE FOUND TO SUPPORT THIS
559   * 17  ** OBSERVATION IT IS OMITTED FROM THIS DESIGN.
560   * 18  DC CASE IACM31:3) ..FORM NEW Y AND S_DESIGNATOR ACCORDING TO IACM.
561   * 19  ** 'B'CODE' = L(Y)
562   * 20  ** S_DESIGNATOR = U(S)
563   * 21  ** 'B'CODE'\Y = U(S) + B(U(S))15)
564   * 22  ** S_DESIGNATOR = B(U(S))15,3)
565   * 23  ** ELSE Y = U(Y) + B(U(Y))15,
566   * 24  ** S_DESIGNATOR = U(S)
567   * 25  END CASE
568   * 26  ENDCASE ..END WHILE
569   * 27  C = IACM(C) ..RETURN DATA STRUCTURE INFORMATION TO CALLER.
570   * 28  CI = IACM(C).1 ..SAVE BIT 29 IN CASE =0, FMT IP, C IWS OR IWS.
571   * 29  IE_C = 0C AND. REPEAT IN PROCESS "REPEAT OF SPECIAL INDIRECT", THEN
572   * 30  CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-2,CP ILLEGAL INSTRUCTION") ..ABORT THE INSTRUCTION.
573   * 31  EDIE
574   * 32  IE_C = CC AND. UCF1=025 ..SB ILLEGAL IF I=1 AND C=0"-. THEN
575   * 33  CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P-1,CP ILLEGAL INSTRUCTION") ..ABORT THE INSTRUCTION.
576   * 34  EDIE
577   * 35  IE_C GCD "CHARACTER REFERENCE", THEN
578   * 36  P := IACM(P)
579   * 37  P&Y = IACM(Y) 1 BITS RIGHT JUSTIFIED IN A 32-BIT WORD
580   * 38  SET CHARACTER_ADDRESSING_OVERRIDE
581   * 39  IE SEGMENTAL CHARACTER ADDRESSING - AND. NOT WORD REFERENCE. THEN
582   * 40  ** MODIFY IACM(P) AND MAYBE IACM(Y).
583   * 41  ** IF P-IACM(Y) < 0, THEN ** MUST UPDATE IACM(Y)
584   * 42  ** [IACM(P)] = 32 - IACM(W); IACM(Y) = IACM(Y) + 1
585   * 43  ELSE
586   * 44  ** IACM(P) = IACM(P) - IACM(Y)
587   * 45  EDIE
588   * 46  EDIE
589   * 47  EDIE
590   * 48  BEQUEN

```

NSAC  
אנו מודים לך על תרומותך  
לפזון צדקה (בגדי חיל)

MEMORY\_BLOCK 1-8-BE5, DEF 32-BE0, S\_DESIGNATOR, RESUME\_TYPE, P\_MCDI

卷之三

SPR\_CHECK (S\_INDICATOR, CHECK\_TYPE, P\_MODIFICATION, 16\_REG)

```

REF PAGE *****
618   * 1   .. IF PROTECTION CHECKING IS ENABLED (AS DETERMINED IN MAIN LOGIC),
619   * 2   .. THIS ROUTINE PERFORMS THE ACCESS TYPE CHECK
620   * 3   .. DESIGNATE BY CALLER AND THE DISPLACEMENT CHECK. ON
621   * 4   .. ANY PROTECTION VIOLATION P IS MODIFIED AS INDICATED, THE
622   * 5   .. APPROPRIATE INTERRUPT GENERATED, AND THE INSTRUCTION
623   * 6   .. IS ABORTED. SOFTWARE PROTECTION REGISTER (SPR) FIELDS (E.G., IA, R,
624   * 7   .. IP, ETC.) ARE AS DEFINED ON THE READER/IRE CARD.
625   * 8   .. (1) S_INDICATOR - BASE REGISTER ASSOCIATED WITH THE ADDRESS
626   * 9   .. (2) CHECK_TYPE - SPECIFIES WHETHER OPERAND, INSTRUCTION OR INDIRECT ADDRESSING TYPE
627   * 10  .. (3) P_MODIFICATION - P REGISTER MODIFICATION IF AN ERROR OCCURS
628   * 11  .. (4) 16_REG - VALUE TO BE COMPARED AGAINST SPR LIMIT FIELD
629   * 12  .. IF SPR_CHECKS ENABLED, THEN
630   * 13  ..   FETCH APPROPRIATE SPR (..E.G., CMR(10160+5 INDICATOR))
631   * 14  .. IF CHECK_TYPE VALID FOR THIS SPR, THEN
632   * 15  ..   IF CHECK_TYPE = IA "INDIRECT ADDRESSING, THEN
633   * 16  ..     IF SPR_PRIVILEGED_INSTRUCTION_SET (.AND. SPEC1), THEN
634   * 17  ..       INST(I) SET AND SPR(IR) SET
635   * 18  ..       ..SPECIAL PRIVILEGED INSTRUCTION(IT ON REP CARD) => INST(I) SET AND SPR(IR) SET
636   * 19  ..       CALL GENERATE_SYNCHRONOUS_INTERRUPT(P_MODIFICATION, PRIVILEGED INSTRUCTION)
637   * 20  ..       ..ABORT THE INSTRUCTION.
638   * 21  .. ELSE
639   * 22  ..   IF SPEC1(.2) (.AND. SPEC1) = 0, THEN
640   * 23  ..     ..PROTECTION VIOLATION, ATTEMPT TO USE INTERRUPT & S REGISTERS
641   * 24  ..     ..FOR INDIRECT REFERENCE WITHOUT AUTHORIZATION.
642   * 25  ..     CALL GENERATE_SYNCHRONOUS_INTERRUPT (P_MODIFICATION, IA OR OP READ),
643   * 26  ..     ..ABORT THE INSTRUCTION.
644   * 27  .. ENDIF
645   * 28  .. ENDIF
646   * 29  .. IF 16_REG(SPR), THEN
647   * 30  ..   GENERATE_SYNCHRONOUS_INTERRUPT ("P-- P_MODIFICATION, "APPROPRIATE LIMIT INTERRUPT")
648   * 31  ..   ..ABORT CURRENT INSTRUCTION.
649   * 32  .. ENDIF
650   * 33  .. ELSE
651   * 34  ..   ..THERE IS A PROTECTION VIOLATION.
652   * 35  ..   GENERATE_SYNCHRONOUS_INTERRUPT ("P-- P_MODIFICATION, "APPROPRIATE TYPE VIOLATION")
653   * 36  ..   ..ABORT CURRENT INSTRUCTION.
654   * 37  .. ENDIF
655   * 38  .. ENDIF
656   * 39  .. RETURN

```

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 21

\* CP INSTRUCTION SUPPORT ROUTINES

## CP/IOC\_CLOCK\_COMMUNICATIONS

PAGE

```
663 * 1 ** THIS ROUTINE CAN BE USED BY THOSE CP INSTRUCTIONS REQUIRING A
664 * 2 ..RESPONSE (VIA THE Q BUS) FROM THE IOC. AFTER INITIATING A REQUEST
665 * 3 ..TO THE IOC, THE FUNCTION DESIGNATORS ARE SENT TO THE IOC ACROSS THE
666 * 4 ..Q BUS. THE IOC THEN RESPONDS WITH 32 BITS ON THE Q BUS. THIS RESPONSE
667 * 5 ..WILL BE AVAILABLE TO THE CALLING INSTRUCTION.
668 * 6 INITIATE A REQUEST TO THE IOC DESIGNATED BY U(1A)
669 * 7 ** ENSURE THAT THIS REQUEST HAS BEEN ACCEPTED.
670 * 8 Q_BUSETC3L6) := U(6F) **SEND FUNCTION DESIGNATORS FROM U.
671 * 9 Q_BUSETC2,3) := U(CE2) **TO INCLUDE SUBFUNCTION DESIGNATORS.
672 * 10 Q_BUSETC2) := 0 **ENSURE THAT THIS REQUEST NOT INTERPRETED AS AN INTERRUPT STATUS CODE REQUEST.
673 * 11 SEND Q_BUS TO THE IOC
674 * 12 WAIT UNTIL IOC HAS ACQUIRED THE INFORMATION FROM THE Q_BUS
675 * 13 RECEIVE THE IOC'S RESPONSE ON THE Q_BUS
676 * 14 RETURN ..INFORMATION NOW AVAILABLE ON Q BUS.
```

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SUPPORT ROUTINES

DU\_JUMP (S\_DESIGNATOR, \_DISPLACEMENT)

REF  
PAGE

679 \* 1 \*\* ROUTINE USED BY JUMP TYPE INSTRUCTIONS TO UPDATE THE PROGRAM COUNTER TO THE ADDRESS  
680 \* 2 .. OF THE "JUMPED TO" INSTRUCTION. CALLED WHEN A JUMP IS TO BE "TAKEN".  
681 \* 3 .. (1) S\_DESIGNATOR - BASE REGISTER INDICATOR FOR NEW PROGRAM COUNTER (P(S)).  
682 \* 4 .. (2) \_DISPLACEMENT - DISPLACEMENT FOR NEW PROGRAM COUNTER (P(O)).  
683 \* 5 P(S) == S\_DESIGNATOR  
684 \* 6 P(O) == \_DISPLACEMENT  
685 \* 7 RETRN

NSWC AN/UVK-7 (CP)  
CP INSTRUCTION SUPPORT ROUTINES

## GET\_SHIFT\_AMOUNT (\_M, REF SHIFT\_COUNT)

```

REF PAGE *****
689   *   1   ** RETURN SHIFT COUNT INTERPRETED FROM M FIELD OF FAT IV & INSTRUCTION.
690   *   2   ** (1) M - M FIELD OF FAT IV & INSTRUCTION
691   *   3   ** (2) SHIFT_COUNT - SHIFT COUNT RETURNED TO CALLER
692   *   4   IF _M(6,1) = 0, THEN
693   *   5   SHIFT_COUNT = _M(5)
694   *   6   ELSE
695   *   7   IF _M(5,1) = 0, THEN
696   *   8       CALL GET_AREG (_M(3,3), 32_REG(1))
697   *   9       SHIFT_COUNT = 32_REG(1)(7) . . . 6 BITS OF COUNT FOR DOUBLE SHIFTS.
698   * 10   ELSE
699   * 11       CALL GET_AREG (_M(3,3), 32_REG(1))
700   * 12       SHIFT_COUNT = 32_REG(1)(5) . . . 6 BITS OF COUNT FOR DOUBLE SHIFTS.
701   * 13   ENDIF
702   * 14   ENDIF
703   * 15   RETURN

```

## HALF-WORD\_TOGGLE

REF  
PAGE

```
    * 1   ** UPDATE UPPER/LOWER DESIGNATOR (ASR(15,1)) UPON COMPLETION OF A HALF-WORD INSTRUCTION. CALLED
    * 2   ** BY ALL HALF-WORD INSTRUCTIONS AFTER CHECKING FOR SYNCHRONOUS INTERRUPTS. THIS ALLOWS
    * 3   ** THE RIGHI ASR TO BE STOPPED UPON INTERRUPT (ECP-34).
    * 4   ** NOTE: A HALF-WORD INSTRUCTION EXECUTED REMOTELY DOES NOT AFFECT THE UPPER/LOWER DESIGNATOR.
    * 5   IE .NOT. EXECUTE_REMOTE_IN_PROGRESS, THEN
    * 6   IF ASR(15,1), THEN
    * 7     ASR(15,1) = 0 .NEXT INSTRUCTION FROM UPPER HALF.
    * 8   ELSE
    * 9     IF ULC(F) = 0'60' "ASSUME 2 HALF-WORD INSTRUCTIONS", THEN
    * 10      ASR(15,1) = 1 .EXECUTE LOWER HALF NEXT.
    * 11   ENDIF
    * 12   ENDIF
    * 13   ENDIF
    * 14   RETURN
```

ANSWER-7 [CP]  
CP INSTRUCTION SUPPORT ROUTINES

14 DEC 79 PAGE 26

REPLACE\_CHECK (32\_REG)

REF  
PAGE \*\*\*\*\*  
\* 1     \*\* IF THE CURRENT INSTRUCTION IS A REPLACE TYPE, THEN UPDATE THE MEMORY  
\* 2     \*\* LOCATION (Y) SPECIFIED BY THE OPERAND ADDRESS.  
\* 3     \*\* (1) 32\_REG - 32 BIT VALUE TO BE STORED IN MEMORY  
\* 4     \*\* IF (CF) >= 03, THEN \*\*THIS IS A REPLACE TYPE INSTRUCTION.  
\* 5     CALL \_REPLACE (32\_REG)  
\* 6     ENDIF  
\* 7     RETURN  
\* 8

MSWC      A/W/YK-7 (CP)      CP INSTRUCTION SUPPORT ROUTINES

-REPLACE\_132\_REG()

REF PAGE	1	2	3	4	5	6	7	8	9	10	11	12	13
731	• STORE 32_REG INTO THE MEMORY LOCATION (Y) SPECIFIED BY THE OPERAND												
732	• ADDRESS. THE ALGORITHM USED TO COMPUTE THIS ADDRESS DIFFERS WHEN A	• REPEAT IS IN PROGRESS.											
733	• 3 •	• (1) 32_REG - 32_BIT VALUE TO BE STORED IN MEMORY											
734	• 4	• IF REPEAT_IN_PROGRESS = AND. REPEAT_ACB) <> 0, THEN											
735	• 5	• SAVE_UCS)											
736	• 6	• UCS) != 0 • FORCE SO ON STORE FOR REPEATED INSTRUCTIONS.											
737	• 7	• CALL_OP_STORE(132_REG, "DISPLACE=0) • 32_REG -> MEMORY											
738	• 8	• RESTORE_UCS)											
739	• 9	• ELSE											
740	• 10	• CALL_OP_STORE(132_REG, "DISPLACE=0) • 32_REG -> MEMORY											
741	• 11	• ENDIF											
742	• 12	• RETURN											
743	• 13												

MSMC AN/UYK-7 (CP)  
CP INSTRUCTION SUPPORT ROUTINES

UPDATEA\_REPLACE (32\_REG, A\_DESIGNATOR)

16 DEC 79 PAGE 28

REF	PAGE	1	• UPDATE THE ACCUMULATOR A_DESIGNATOR AND PERFORM REPLACE INSTRUCTION
745	*	2	• CHECKS.
746	*	3	• (1) 32_REG - 32 BIT REGISTER CONTAINING VALUE TO BE STORED
747	*	4	• (2) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)
748	*	5	• REGISTER TO BE UPDATED
749	*	6	• CALL PUT_AREG (A_DESIGNATOR, 32_REG)
750	*	7	• CALL REPLACE_CHECK (32_REG)
751	*	8	• RETURN
752	*		

NSMC  
ANSWER-7 CCP  
CP INSTRUCTION SUPPORT ROUTINES

SET\_C01 (QUANTITY1, QUANTITY2)

REF	PAGE	
754	1	** COMPARE QUANTITY1 TO QUANTITY2 SETTING EQUAL/INEQUAL AND GREATER OR EQUAL/LESS THAN
756	2	** ASR(2,2) ACCORDINGLY.
757	3	** (11) QUANTITY1 - 32-BIT VALUE.
758	4	** (21) QUANTITY2 - 32-BIT VALUE.
759	5	IF QUANTITY1 = QUANTITY2, THEN
760	6	ASR(2,1) := 1 ..SET EQUAL INDICATION.
761	7	ELSE ASR(2,1) := 0 ..SET NOT EQUAL INDICATION.
762	8	ENABLE
763	9	IF QUANTITY1 >= QUANTITY2, THEN
764	10	ASR(1,1) := 1 ..SET GREATER OR EQUAL INDICATION.
765	11	ELSE
766	12	ASR(1,1) := 0 ..SET LESS THAN INDICATION.
767	13	ENABLE
768	14	REINH
769	15	

SET\_CD2 (QUANTITY1, QUANTITY2, QUANTITY3)

REF PAGE \*\*\*\*\*  
771 \* 1 .. COMPARE QUANTITY1 TO QUANTITY2 AND QUANTITY3 SETTING OUTSIDE/WITHIN LIMITS  
773 \* 2 .. (ASR(0,1)) ACCORDINGLY.  
774 \* 3 .. (1) QUANTITY1 - 32-BIT REGISTER.  
775 \* 4 .. (2) QUANTITY2 - 32-BIT REGISTER.  
776 \* 5 .. (3) QUANTITY3 - 32-BIT REGISTER.  
777 \* 6 .. IF QUANTITY3 > QUANTITY1 AND QUANTITY2 >= QUANTITY3, THEN  
778 \* 7 .. ASR(0,1) := 0 ..SET WITHIN LIMITS INDICATION.  
779 \* 8 .. ELSE ..ASR(0,1) := 1 ..SET OUTSIDE LIMITS INDICATION.  
780 \* 9 ..  
781 \* 10 .. ENDIF  
792 \* 11 .. RETURN  
\*\*\*\*\*

M5HC

AN/UYK-7 (CP)

14 DEC 79 PAGE 31

\* CP UTILITY ROUTINES \*

## ADD\_S (S\_DESIGNATOR, REF Y)

REF

PAGE

```
*****  
* 1   .. ADD THE APPROPRIATE BASE(S) REGISTER TO THE DISPLACEMENT VALUE(Y).  
* 2   .. (11) S_DESIGNATOR - INTEGER NUMBER SPECIFYING THE BASE(S) REGISTER  
* 3   .. (12) SUPPLY INFORMATION USED MODULO 8).  
* 4   .. (21) Y - AN 16-BIT QUANTITY CONTAINING A ZERO EXTENDED 16-BIT VALUE (E.G. U(Y)+B1U(B2)).  
* 5   ..UPON ENTRY, THE 16-BIT RESULT OF THE ADDITION WILL BE RETURNED HERE.  
* 6   ..NOTE: THE BASE REGISTER IS NOT ADDED ON MDR REFERENCES.  
* 7   ..IF S_DESIGNATOR <> 7 .JP. ASR(7,1)=0 "NOT MDR REFERENCE", THEN  
* 8   ..CALL GET_SREG (PCOULD(S_DESIGNATOR,8), 32_REG(1)) ..GET DESIRED BASE REGISTER.  
* 9   ..Y = Y + 32_REG(1) ..FORM 16-BIT VALUE.  
* 10  ..ENDIF  
* 11  ..RETURNS  
* 12  ..  
*****
```

NS/C  
AN/UYK-7 (CP)  
CP UTILITY ROUTINES

14 DEC 79 PAGE 33

GET\_AREG IA\_DESIGNATOR, REF 32\_REG;

```
REF PAGE *****
799   * 1    ** RETURN THE CONTENTS OF ACCUMULATOR IN 32_REG.
E00   * 2    ** (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)
601   * 3    ** REGISTER TO SUPPLY INFORMATION (USED MODULO 8)
E02   * 4    ** (2) 32_REG - 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING
603   * 5    ** ACCUMULATOR VALUE
604   * 6    ** A_DESIGNATOR == MODULUS(A_DESIGNATOR, 8)
605   * 7    ** IF ASP(10,1) == 1/B REGISTER SELECT" THEN
606   * 8    ** 32_REG == CHR100+A_DESIGNATOR) .INTERRUPT ACCUMULATOR SELECTED.
607   * 9    ELSE
608   * 10   32_REG := CHR(0+A_DESIGNATOR) .TASK ACCUMULATOR SELECTED.
609   * 11   ENDIF
610   * 12   RETURN
* ****
```

ANSWER-7 (CF)  
CP UTILITY ROUTINES

14 DEC 79 PAGE 34

GET\_BREG (B\_DESIGNATOR, REF 32\_REG)

REF PAGE

```
* 1   * LEAD THE LOW ORDER 16-BITS OF 32_REG WITH THE CONTENTS OF THE APPROPRIATE INDEX (B) REGISTER.
* 2   * (1) B_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (B) REGISTER.
* 3   * (2) 32_REG - 32-BIT REGISTER (USED IFINDEX B).
* 4   * JUSTIFIED 16-BIT QUANTITY FROM THE SPECIFIED INDEX REGISTER.
* 5   * B_DESIGNATOR := MCWULC(B_DESIGNATOR, 8)
* 6   *
* 7   * E B_DESIGNATOR = 0, THEN
* 8   * 32_REG(16) := 0 ..B10; IS A SOURCE OF ZEROS.
* 9   *
* 10  * ELSE
* 11  * IF ASR(10,1) "A/B REGISTER SELECT", THEN
* 12  *   32_REG(16) := CMR(110+B_DESIGNATOR) ..I.E. APPROPRIATE INTERRUPT INDEX (B) REGISTER.
* 13  * ELSE
* 14  *   32_REG(16) := CMR(10+B_DESIGNATOR) ..I.E. APPROPRIATE TASK INDEX (B) REGISTER.
* 15  * ENDIF
* 16  * BEGEND
```

## GET\_SREG\_IS\_DESIGNATOR, REF 32\_REG

REF PAGE \*\*\*\*\*

```
*   1   ** RETURN THE CONTENTS OF THE BASE REGISTER S_DESIGNATOR IN 32_REG.
632 *   2   ..(1) S_DESIGNATOR = BASE REGISTER DESIGNATOR
633 *   3   ..(2) 32_REG = 32 BIT REGISTER (SUPPLIED BY CALLER) RECEIVING
634 *   4   ..BASE REGISTER CONTENTS.
635 *   5   S_DESIGNATOR := MODULO(S_DESIGNATOR, 8);
636 *   6   IF ASR(1,1) ==BASE(1) REGISTER SELECT", THEN
637 *   7   32_REG := CHAR120 + S_DESIGNATOR) ..INTERRUPT BASE REGISTER SELECTED.
638 *   8   ELSE
639 *   9   32_REG := CHAR120 + S_DESIGNATOR) ..TASK BASE REGISTER SELECTED.
640 *   10  ENDIF
641 *   11  RETURN
642 *   12
643 *   13
644 *   14
```

MSWC AM/UTK-7 (CP)  
CP UTILITY ROUTINES

14 DEC 79 PAGE 36

PUT\_REG (A\_DESIGNATOR, 32\_REG)

REF PAGE \*\*\*\*\*

```
846 * 1  ** STORE THE CONTENTS OF 32_REG INTO THE ACCUMULATOR A_DESIGNATOR.  
847 * 2  ** (1) A_DESIGNATOR - INTEGER NUMBER SPECIFYING THE ACCUMULATOR (A)  
848 * 3  ** REGISTER TO BE UPDATED (USED RORBLD B)  
849 * 4  ** (2) 32_REG - 32 BIT VALUE TO BE STORED.  
850 * 5  A_DESIGNATOR := RORBLD(A_DESIGNATOR, 0)  
851 * 6  IF ASR10(1) "A/B REGISTER SELECT" THEN  
852 * 7  CHN(100+A_DESIGNATOR) := 32_REG //INTERRUPT ACCUMULATOR SELECTED.  
853 * 8  
854 * 9  CHN(0+A_DESIGNATOR) := 32_REG // TASK ACCUMULATOR SELECTED.  
855 * 10 ENDIF  
856 * 11 RETURN
```

NSUC AN/UYK-7 (CP)  
C7 UTILITY ROUTINES

14 DEC 79 PAGE 37

PUT\_BREG (B\_DESIGNATOR, 32\_REG)

REF PAGE

```
858      * 1    ** STORE THE LOWER 19 BITS OF 32_REG INTO THE APPROPRIATE INDEX REGISTER.  
859      * 2    ** (1) B_DESIGNATOR - INTEGER NUMBER SPECIFYING THE INDEX (B) REGISTER  
860      * 3    ** TO BE UPDATED MODULO 81.  
861      * 4    ** (2) 32_REG - 32-BIT REGISTER CONTAINING RIGHT JUSTIFIED 19-BIT  
862      * 5    ** QUANTITY TO BE STORED INTO THE SPECIFIED INDEX REGISTER.  
863      * 6    B_DESIGNATOR == MODULUS B_DESIGNATOR, B1  
864      * 7    IF B_DESIGNATOR > 0, THEN ..ATTEMPT TO STORE INTC B10 IS A NOOP.  
865      * 8    IF ASR(10,1) "A/B REGISTER SELECT", THEN  
866          CMR(110+B_DESIGNATOR) := 32_REG(10) .DEFINE INTERRUPT INDEX (B) REGISTER.  
867      * 9  
868      * 10  
869      * 11    ELSE  
870          CMR(10+B_DESIGNATOR) := 32_REG(10) ..DEFINE TASK INDEX (B) REGISTER.  
871      * 12    ENDIF  
872      * 13    ENDIF  
873      * 14    RETURN
```

NSWC            AN/UYK-7 (CP)  
                  CP UTILITY ROUTINES

14 DEC 79 PAGE 38

PUT\_SREG (S\_DESIGNATOR, 32\_REG)

BEF PAGE \*\*\*\*\*  
\*\*\*\*\*  
875 \* 1        \* STORE THE CONTENTS OF 32\_REG INTO THE BASE REGISTER S\_DESIGNATOR.  
876 \* 2        \* (1) S\_DESIGNATOR = BASE REGISTER DESIGNATOR  
877 \* 3        \* (2) 32\_REG = 32 BIT REGISTER CONTAINING VALUE TO BE STORED  
878 \* 4        \* S\_DESIGNATOR = MODULUS\_DESIGNATOR, 0  
879 \* 5        \* IF ASR11(1) = BASE(S) REGISTER SELECT, THEN  
            \*     CR1120 + S\_DESIGNATOR) := 32\_REG --INTERRUPT BASE REGISTER SELECTED.  
880 \* 6        \* ELSE  
881 \* 7        \*     CR1120 + S\_DESIGNATOR) := 32\_REG --TASK BASE REGISTER SELECTED.  
882 \* 8        \* ENDIF  
883 \* 9        \* RETURNS  
884 \* 10      \*  
\*\*\*\*\*

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 39

\*\*\*\*\*  
\* FLOWING POINT SUBROUTINES \*  
\*\*\*\*\*

## FLOATING\_ADD\_SUBTRACT\_MEAGER (REF M\_MANTISSA, REF A\_MANTISSA)

```

REF PAGE ****
  *   1   ..(1) M_MANTISSA - 64 BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM MEMORY.
  *   2   ..(2) A_MANTISSA - 64-BIT REGISTER SUPPLIED BY CALLER TO CONTAIN MANTISSA FROM A(U(A)+1).
  *   3   ..
  *   4   SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND UC1) SET.
  *   5   CALL OP_READ (32_REG1), "DISPLACEMENT"01 ..GET CHARACTERISTIC FROM MEMORY.
  *   6   CALL OP_READ (M_MANTISSA[63,32]), "DISPLACEMENT"1 ..GET MANTISSA FROM MEMORY.
  *   7   EXTEND SIGN BIT INTO M_MANTISSA[31,32]
  *   8   CALL GET_ARC (U(A)), 32_REG6211 ..GET CHARACTERISTIC FROM A(U(A)).
  *   9   CALL GET_ARC (U(C)+1, A_MANTISSA[63,32]) ..GET MANTISSA FROM A(U(A)+1).
  * 10   EXTEND SIGN BIT INTO A_MANTISSA[31,32]
  * 11   IF M_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN
      * 32_REG11 := 32_REG121 ..ASSUME INTERMEDIATE CHARACTERISTIC THAT OF ACCUMULATOR VALUE.
  * 12   ENDIF
  * 13   IF A_MANTISSA=ZERO "NEGATIVE OR POSITIVE", THEN
  * 14   32_REG12 := 32_REG11 ..ASSUME INTERMEDIATE CHARACTERISTIC THAT OF MEMORY VALUE.
  * 15   ENDIF
  * 16   ENDIF
  * 17   SHIFT COUNT := 32_REG12)-32_REG11 ..ONES COMPLEMENT SUBTRACT.
  * 18   IF SHIFT-COUNT > Q, THEN
      * 19   SHIFT M_MANTISSA RIGHT ARITHMETIC SHIFT_COUNT PLACES
  * 20   ELSE
      * 21   IF SHIFT-COUNT <> 0 "POSITIVE OR NEGATIVE", THEN
          * 22   SHIFT_COUNT := .NOT. SHIFT_COUNT ..ONES COMPLEMENT THE COUNT.
          * 23   SHIFT A_MANTISSA RIGHT ARITHMETIC BY SHIFT_COUNT PLACES
  * 24   ENDIF
  * 25   CALL PUT_ARC (U(A), 32_REG11) ..UPDATE INTERMEDIATE CHARACTERISTIC IN A(U(A)).
  * 26   ENDIF
  * 27   RETURN ..INTERMEDIATE CHARACTERISTIC LEFT IN A(U(A)) FOR LATER USE!

```

NSWC

AHUYK-7 (CP)  
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 41

FLOATING\_OVERFLOW REF 64\_REG, REF 32\_REG;

REF  
PAGE

```
926   * 1    *(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING THE INTERMEDIATE RESULT OF A
      * 2    **FLOATING POINT OPERATION AFTER OVERFLOW HAS OCCURRED.
      * 3
      * 4    SHIFT 64_REG RIGHT 1 BIT **DIVIDE BY 2.
      * 5    64_REG(63,1) := NOT(64_REG(63,1)) **RESTORE SIGN BIT.
      * 6    32_REG := 32_REG +1 **INCREMENT THE CHARACTERISTIC.
      * 7    RETURN
```

## FLOATING\_NORMALIZE (REF 64\_REG, REF 32\_REG)

REF  
PAGE \*\*\*\*\*  
\$36 \* 1 -- (1) 64\_REG - 64-BIT REGISTER SUPPLIED BY CALLER TO BE NORMALIZED AND STORED IN A(U(A))  
\$38 \* 2 32\_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING INTERMEDIATE CHARACTERISTIC.  
940 \* 3 IF 64\_REG(31) = "NEGATIVE", THEN  
941 \* 4 SHIFT\_COUNT := 16 OF LEFT JUSTIFIED 1 BITS IN 64\_REG) - 1  
942 \* 5 ELSE  
943 \* 6 SHIFT\_COUNT := 16 LEFT JUSTIFIED 0 BITS IN 64\_REG) - 1  
944 \* 7 ENDIF  
945 \* 8 IF SHIFT\_COUNT < 0"77", THEN  
946 \* 9 SHIFT LEFT CIRCULARLY 64\_REG BY SHIFT\_COUNT  
947 \* 10 32\_REG := 32\_REG - SHIFT\_COUNT ..ADJUST CHARACTERISTIC (ONES COMPLEMENT)  
949 \* 11 ELSE  
950 \* 12 32\_REG := 0  
951 \* 13 ENDIF  
952 \* 14 PLEIB

NS/C  
FLOORING POINT SUBROUTINES

14 DEC 79 PAGE 43

FLOORING\_ROUND (REF\_C6\_REG, REF\_32\_REG)

```
REF  
PAGE *****  
* 1      **(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING RANTISSA TO BE ROUNDED.  
* 2      **(2) 32_REG - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.  
* 3      IF C6_REG[31,1]>64_REGS[31] "SIGN BIT", THEN  
* 4          IF 64_REGS[31] = "NEGATIVE", THEN  
* 5              64_REGS[3,32] := 64_REGS[63,32]-1  
* 6          ELSE  
* 7              64_REGS[3,32] := 64_REGS[63,32]+1  
* 8          ENDIF  
* 9          IF OVERFLOW OCCURRED, THEN  
* 10              CALL FLOATING_OVERFLOW (64_REG, 32_REG)  
* 11          ENDIF  
* 12      ENDIF  
* 13  PENDON
```

NSWC AN/UYK-7 (CP)  
FLOATING POINT SUBROUTINES

14 DEC 79 PAGE 44

ROUND\_UP (REF MANTISSA, REF CHARACTERISTIC)

REF  
PAGE

\* 1 ..(1) MANTISSA - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING MANTISSA TO BE ROUNDED UP.  
\* 2 ..(2) CHARACTERISTIC - 32-BIT REGISTER SUPPLIED BY CALLER CONTAINING CHARACTERISTIC.  
\* 3 MANTISSA := MANTISSA+1  
\* 4 IF MANTISSA<(31'-1) "OVERFLOW OCCURRED, THEN  
\* 5 ..NOTE THAT WORKING ONLY WITH POSITIVE QUANTITIES.  
\* 6 MANTISSA := MANTISSA.RL-1 ..ADJUST THE MANTISSA.  
\* 7 CHARACTERISTIC := CHARACTERISTIC+1 ..UPDATE THE CHARACTERISTIC.  
\* 8 ENDIF  
\* 9 RETURN  
\* 0

NSWC

AN/UYK-7 (CP)  
FLOATING POINT SUBROUTINES

DIVICE\_COMPARE (REF 64\_REG, 32\_REG)

REF

PAGE

```
982   *    1    ..(1) 64_REG - 64-BIT REGISTER SUPPLIED BY CALLER CONTAINING DIVIDEND.  
983   *    2    ..(2) 32_REG - 32-BIT REGISTER CONTAINING DIVISOR.  
984   *    3    IF 32_REG<=64_REG(63:32), THEN ..USE 32-BIT UNSIGNED COMPARE.  
985   *    4    64_REG(63:32) := 64_REG(63:32) - 32_REG ..ONES COMPLEMENT SUBTRACT.  
986   *    5    t4_REG(0,1) := 1  
987   *    6    ENDIF  
988   *    7    RETURN
```

14 DEC 79 PAGE 45

NSWC AN/UYK-7 (CP)  
FLOATING POINT SUBROUTINES

FLOATING\_POINT\_ENC (MANTISSA, CHARACTERISTIC)

REF  
PAGE

```
*****  
990 * 1  ** COMMON CODE FOR TERMINATION OF FLOATING POINT INSTRUCTIONS.  
991 * 2  ..(1) MANTISSA - 32-BIT RESULTANT MANTISSA  
992 * 3  ..(2) CHARACTERISTIC - 32-BIT RESULTANT CHARACTERISTIC.  
993 * 4  IF MANTISSA=0 "RESULT ZERO", THEN  
994 * 5  CHARACTERISTIC = 0 --ZERO THE CHARACTERISTIC.  
995 * 6  ENDIF  
996 36 * 7  CALL PUT_AREG (U(L)+1, MANTISSA)  
997 36 * 8  CALL PUT_AREG (U(A), CHARACTERISTIC)  
998 * 9  IF (CHARACTERISTIC(L+1))>CHARACTERISTIC(L), THEN  
999 11 * 10  CALL GENERATE_SYNCHRONOUS_INTERRUPT (*P-R1, FLOATING POINT OVERFLOW)  
1000 11 * 11  ..ABORT THE INSTRUCTION.  
1001 * 12  ENDIF  
1002 * 13  RETURN  
*****
```

14 DEC 79 PAGE 46

HSMC

AN/UYK-7 (CP)

16 DEC 79 PAGE 47

\*\*\*\*\*  
\* CP INSTRUCTION SET \*  
\*\*\*\*\*

N5MC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 48

\_DR(6 \_R(6) ..(REPLACE) INCLUSIVE OR FMT II F=01 D 6 F=03 3

REF

PAGE

1005	14	1	CALL OP READ(32,REG(1))"DISPLACE=0" -ACQUIRE OPERAND (Y) AND ALU(A).
1005	33	2	CALL GET_AREG(U), REPEAT_ACCUMULATOR
1007	*	3	REPEAT_ACCUMULATOR :- REPEAT_ACCUMULATOR -Y- 32_R(6)(1) ..PERFORM INCLUSIVE OR.
1009	28	4	CALL UPDATEA_REPLACE(REPEAT_ACCUMULATOR, U(A)) ..RESULT GOES TO CMP & POSSIBLY TO MEMORY.
1011	*	5	RETURN

NSwC                    AN/UYK-7 (CP)  
                          CP INSTRUCTION SET

14 DEC 79 PAGE 49

-SCIE -RSC) .\*(REPLACE) SELECTIVE CLEAR & FMT III F=01 1 E F=03 1

REF  
PAGE

```
1013     14     1     CALL OP_PED (32_REG(1), "DISPLACE=0) .*(FETCH OPERAND Y) AND A(U(A)).  
1015     *     2     32_REG(1) 1" .NOT. 32_REG(1).COMPLEMENT Y.  
1016     3     CALL GET_REG(U(A)), REPEAT_ACCUMULATOR  
1017     *     4     REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR +A. 32_REG(1)  
1018     28     5     CALL UPDATE_REPLACE (REPEAT_ACCUMULATOR, U(A)) .*(RESULT GOES CMP & POSSIBLY TO MEMORY.  
1020     *     6     RETURN
```

AM/UTK-7 (CP)  
CP INSTRUCTION SET

\_RS1E \_RS1S) .. (REPLACE) SELECTIVE SUBSTITUTE F#1 II F=01 2 E F=03 2

REF  
PAGE

1022 14 \* 1 CALL OPREAD(32\_REG(1), "DISPLACE"0) ..FETCH OPERAND (Y), A[U(A)], & A[U(A)+1].  
1024 33 \* 2 CALL GET\_AREG(U(A), 32\_REG(2))  
1025 33 \* 3 CALL GET\_AREG(U(Y+1), REPEAT\_ACCUMULATOR)  
1026 \* 4 32\_REG(1) := 32\_REG(1) - A \* 32\_REG(2)  
1027 \* 5 REPEAT\_ACCUMULATOR := (REPEAT\_ACCUMULATOR \*A, 1,MDI,32\_REG(2))) .V. 32\_REG(1)  
1029 28 \* 6 CALL UPDATE\_REPLACE\_AREGULATOR, U(A)+1 ..RESULT GOES TO A[U(A)+1] & POSSIBLY TO MEMORY.  
1031 \* 7 RETRN

NSWC

AN/UTK-7 (CPI  
CP INSTRUCTION SET

14 DEC 79 PAGE 51

\_XORIE\_RMOP) ..(REPLACE) EXCLUSIVE OR FNI II F-01 3 & F-03 3

REF

PAGE \*

1033 16 \* 1 CALL GP READ(32\_REG[1], "DISPLACE"=01) /\*FETCH OPERAND (Y1) & A[U(A)].  
1035 33 \* 2 CALL GET\_AREG(U(A), REPEAT\_ACCUMULATOR)  
1036 \* 3 REPEAT\_ACCUMULATOR /\* REPEAT\_ACCUMULATOR -X- 32\_REG[1] /\*PERFORM EXCLUSIVE OR.  
1038 28 \* 4 CALL UPDATE\_A\_REPLACE(AREGULATOR, U(A)) /\*RESULT GOES TO A[U(A)] & POSSIBLY TO MEMORY.  
1040 \* 5 RETURN  
\*  
\*\*\*\*\*

NSWC AN/UYK-7 (CF)  
CP INSTRUCTION SET

-ALPIE\_REPLACE) ADD LOG. PBD. FMT II F=01 4 E F=03 4

REF PAGE \*

1042 14 \* 1 CALL OP\_READREPETACCUMULATOR,-DISPLACE(F0) ..FETCH OPERAND(Y), A(U(A)) & A(U(A)+1).  
1044 33 \* 2 CALL GET\_ARC((A)), 32\_REG(21)  
1045 33 \* 3 CALL GET\_ARC((A)+1), 32\_REG(31)  
1046 \* 4 REPEAT\_ACCUMULATOR \* (REPEAT\_ACCUMULATOR \*A. 32\_REG(21) + 32\_REG(31)  
1047 \* 5 UPDATE\_FIXED\_POINT\_OVERFLOW(ASR(3,1))  
1048 28 \* 6 CALL UPDATE\_A\_REPLACE(REPEAT\_ACCUMULATOR, U(A)+1) ..RESULT GOES TO CAR & POSSIBLY TO MEMORY.  
1050 \* 7 RETURN  
\* \*\*\*\*\*

14 DEC 79 PAGE 52

NSWC  
ANUVAK-7 (CPI)  
CP INSTRUCTION SET

14 DEC 79 PAGE 53

\_LLP (\_NLP, \_LLPN, \_RLP, E \_RNLPI)

REF

```

PAGE   *   1   .. LOAD, SUBTRACT, AND REPLACE LOGICAL PRODUCT.
1052   *   2   ..FPT 11, F=02 5, Q1 6, 01 7, 03 5, 03 6
1053   *   3   CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0")
1054   *   4   CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0")
1055   *   5   CALL GET_AREG (U(A)), 32_REG(13);
1056   *   6   REPEAT_ACCUMULATOR := 32_REG(3) .A. REPEAT_ACCUMULATOR ..FORM LOGICAL PRODUCT.
1057   *   7   IF _NLP .OR. _RNLPI, THEN
1058   *   8   CALL GET_AREG (U(A))+1, 32_REG(12)
1059   *   9   REPEAT_ACCUMULATOR := 32_REG(12) - REPEAT_ACCUMULATOR ..ONES COMPLEMENT SUBTRACT.
1060   *   10  UPDATE FIXED POINT OVERFLOW BIT IN AS(PBIT 3)
1061   *   11  ENDIF
1062   *   12  IF _LLP, THEN
1063   *   13  CALL PUT_AREG (U(A)), REPEAT_ACCUMULATOR
1064   *   14  ELSE
1065   *   15  CALL UPDATE_REPLACE(REPEAT_ACCUMULATOR, U(A)+1)
1066   *   16  ENDIF
1067   *   17  RETURN
1068   *   18
1069   *   19

```

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

16 DEC 79 PAGE 54

\_CNT • COUNT ONES, FMT II, F=02 0

REF PAGE	1071	14	1	CALL OPREAD (32_REG12), "DISPLACE"0
	1072	*	2	REPEAT_ACCUMULATOR := 0
	1073	*	3	I := 0
	1074	*	4	DO WHILE I < 32 . . .LOOP AND COUNT ONES.
	1075	*	5	IF 32_REG12(I,1), THEN
	1076	*	6	REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 1
	1077	*	7	ENDIF
	1078	*	8	I := I + 1
	1079	*	9	ENDDO
	1080	36	10	CALL PUT_AREG (U1A), REPEAT_ACCUMULATOR, --STORE RESULTS
	1081	*	11	RETURN

NSWC CP INSTRUCTION SET

14 DEC 79 PAGE 95

\_XR (-XRL) ..EXECUTE REMOTE (LOWER). FMT 11. F=02 2. F=02 3

REF PAGE

```
1083   * 1   ** XR EXECUTES ONE INSTRUCTION ONLY 11. E. ONE FULL WORD INSTRUCTION
1084   * 2   ..OR UPPER HALF OF TWO HALF WORD INSTRUCTIONS).
1085   * 3   CALL JUMP_ADDRESS(DISPLACE_0, OPERAND_S, OPERAND_DISPLACEMENT)
1086   * 4   ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE REGISTER
1087   * 5   ..AND OPERAND DISPLACEMENT ARE RETURNED.
1088   * 6   32_REG6(3) = OPERAND_DISPLACEMENT
1089   * 7   CALL ADD_S (OPERAND_S, 32_REG6(3)) ..FIND REMOTE INSTRUCTIONS ABSOLUTE ADDRESS.
1090   * 8   CALL SPI_CHECK (32_REG6(3), INSTRUCTION) ..PERFORM INSTRUCTION BREAKPOINT CHECKS.
1091   * 9   CALL SPI_CHECK (U(5)), INSTRUCTION EXECUTE, "P=P", OPERAND_DISPLACEMENT)
1092   * 10  ..PERFORM INSTRUCTION SPI CHECKS
1093   * 11  QDF2 = U(F2) ..SAVE XR, XRL INDICATOR.
1094   * 12  CALL MEMORY_READ (32_REG6(3), U, UCS3, INSTRUCTION, "P=P")
1095   * 13  ..FETCH REMOTE INSTRUCTION INTO U REGISTER.
1096   * 14  IE OLOF2 ODD, THEN "P" > XRL.
1097   * 15  UU = UL ..COPY LOWER HALF WORD INSTRUCTION TO UU.
1098   * 16  ENDIF
1099   * 17  SET EXECUTE_REMOTE_IN_PROGRESS
1100   * 18  DEFINE INSTRUCTION_FORMAT_INDICATION ..I.E. I, II, III, OR IV.
1101   * 19  CALL _DECODE ..TRANSFER CONTROL TO APPROPRIATE INSTRUCTION.
1102   * 20  RETURN
1103
1104
```

MSFC AN/UVK-7 (CP)  
CP INSTRUCTION SET

\_SLP ..STORE LOGICAL PRODUCT, FMT II, F=02 4

REF	PAGE	
1106	33	1 CALL GET_AREG (U(A), REPEAT_ACCUMULATOR)
1107	33	2 CALL GET_AREG (U(A)+1, 32_REG12)
1108	*	3 REPEAT_ACCUMULATOR = REPEAT_ACCUMULATOR -A. 32_REG12
1109	15	4 CALL OP_STORE (REPEAT_ACCUMULATOR, "DISPLAY=0")
1110	*	5 RETRN

16 DEC 79 PAGE 56

MSIC AN/UYK-7 (CP)  
CP INSTRUCTION SET

-SSUM --STORE SUM, FMT III, F=02 S

REF  
PAGE  
\* 1112 33 \* 1 CALL GET\_AREG (U(A)), AF = AT\_ACCUMULATOR )  
\* 1113 33 \* 2 CALL GET\_AREG (U(A)+1), 32\_REG(21)  
\* 1114 \* 3 REPEAT\_ACCUMULATOR = REPEAT\_ACCUMULATOR + 32\_REG(2) --DINES COMPLEMENT ADDITION.  
\* 1115 \* 4 IF OVERFLOW OCCURRED, THEN  
\* 1116 \* 5 ASR(3,2) = 1 --SET FIXED POINT OVERFLOW INDICATOR.  
\* 1117 \* 6 ELSE  
\* 1118 \* 7 ASR(3,1) = 0 --CLEAR FIXED POINT OVERFLOW INDICATOR.  
\* 1119 \* 8 ENDIF  
\* 1120 36 \* 9 CALL PUT\_AREG ( U(A)+1 ), REPEAT\_ACCUMULATOR;  
\* 1121 10 CALL EP\_STORE (REPEAT\_ACCUMULATOR, -DISPLACEMENT)  
\* 1122 \* 11 BEGEND

16 DEC 79 PAGE 57

MSMC  
A7/UYK-7 (CP)  
CP INSTRUCTION SET

-5015 - STUDIE DIFFERENCE, FNU 11, F-02 6

```

REF PAGE *****
1125   33 * 1 CALL GET_AREG1(A),32_REG11]
1126   33 * 2 CALL GET_AREG1(A)+1,REPEAT_ACCUMULATOR]
1127   33 * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR]
1128   4 * 4 IF OVERFLOW OCCURRED, THEN
1129   5 * 5 ASR3,1] = 1 **SET FIXED POINT OVERFLOW INDICATOR.
1130   6 * 6 ELSE
1131   7 * 7 ASR3,1] := 0 **CLEAR FIXED POINT OVERFLOW INDICATOR.
1132   8 * 8 ENDIE
1133   9 * 9 CALL PUT_AREG1(A)+1,REPEAT_ACCUMULATOR]
1134  10 * 10 CALL OP_STORE(REPEAT_ACCUMULATOR,"DISPLACE=0")
1135  11 * 11 RETURN

```

NSWC

AN/DYK-7 (CP)  
CP INSTRUCTION SET

-DS \*\* DOUBLE STORE ACCUMULATORS, FMT 2I, F=02 7

14 DEC 79 PAGE 59

REF PAGE \*\*\*\*\*  
1137 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION ..INDICATE PRIVILEGED IF SPR{16,11} SET AND UC13 SET.  
1139 \* 2 CALL GET\_AREG(UA),32\_REG(11)  
1140 \* 3 CALL GET\_AREG(UA)+,32\_REG(12)  
1141 \* 4 CALL OP\_STORE(32\_REG(11),"DISPLACE=0")  
1142 \* 5 CALL OP\_STORE(32\_REG(12),"DISPLACE=1")  
1143 \* 6 RETURN  
\*\*\*\*\*

NSwC

AN/UKK-7 (CP)  
CP INSTRUCTION SET

-TSF .. TEST & SET FLAG, FMT II, F=03 7

REF

PAGE

```
1145 * 1 IF UC16,1) "INDIRECT ADDRESSING", THEN --NOT ALIGNED (AS PER REP CARD1).  
1147 11 * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT(-P-M1, CP ILLEGAL INST1) .ABORT THE INSTRUCTION.  
1149 * 3 ENDIF  
1150 14 * 4 CALL OP_REPEAT_ACCUMULATOR, "DISPLACE" 0  
1151 * 5 JE REPEAT_ACCUMULATOR[31:1], THEN  
1152 * 6 ASR(C2,I) := 0 ...NOT EQUAL.  
1153 * 7 ELSE  
1154 * 8 ASR(C2,I) := 1 ..EQUAL.  
1155 * 9 ENDIF  
1156 * 10 REPEAT_ACCUMULATOR[31:1] := 1 ..SET FLAG 611.  
1157 15 * 11 CALL OP_STORE(REPEAT_ACCUMULATOR, "DISPLACE" 0)  
1158 * 12 RETURN
```

14 DEC 79 PAGE 60

NSWC  
ANUWk-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 61

-DL -- DOUBLE LOAD, FMT II F=03 0

REF  
PAGE \*\*\*\*\*  
1160 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION •• INDICATE PRIVILEGED IF SPR[16,1] SET AND U(1) SET.  
1161 \* 2 CALL OP\_FEAR(32\_REG(1), "DISPLACE=0")  
1162 \* 3 SAIL CP\_FEAR(32\_REG(2), "DISPLACE=0")  
1163 \* 4 PUT\_AREG(U(A)>32\_REG(1))  
1164 \* 5 PUT\_AREG(U(A)>1,32\_REG(1))  
1165 \* 6 RETURN  
1166 \*

DA (C DAN) .. DOUBLE ADD &amp; SUBTRACT, FMT II, F=05 1 C F=05 2

REF PAGE \*\*\*\*\*

```
1168   * 1   ** NOTE: 64_REG(1) AND 64_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE LENGTH INSTRUCTION.
1170   * 2   SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(6,1) SET AND UFI) SET.
1172   * 3   CALL OP_READ(64_REG(2)(31,32), "DISPLACE=0"
1173   * 4   CALL OP_READ(64_REG(2)(63,32), "DISPLACE=1")
1174   * 5   GET_AREG(UA), 64_REG(1)(31,32); "DISPLACE=1"
1175   * 6   GET_AREG(UA)+1, 64_REG(1)(63,32)
1176   * 7   JE UCF2) * 1, THEN **ADD.
1177   * 8   64_REG(1) := 64_REG(1) + 64_REG(2) **ONES COMPLEMENT DOUBLE ADD.
1178   * 9   ELSE **SUBTRACT.
1179   * 10  64_REG(1) := 64_REG(1) - 64_REG(2) **ONES COMPLEMENT DOUBLE SUBTRACT.
1180   * 11  ENDIF
1181   * 12  IF OVERFLOW, THEN
1182   * 13  ASR(3,-1) := 1 ..SET FIXED POINT OVERFLOW INDICATOR.
1183   * 14  ELSE
1184   * 15  ASR(3,1) := 0 ..CLEAR FIXED POINT OVERFLOW INDICATOR.
1185   * 16  ENDIF
1186   * 17  PUT_AREG(UA), 64_REG(1)(31,32)
1187   * 18  PUT_AREG(UA)+1, 64_REG(1)(63,32)
1188   * 19  RETURN
*****
```

MSMC      AN/MURK-7 (ICP)  
CP INSTRUCTION SET

-DC .. DOUBLE COMPARE, FAT III, F= 05 3

14 DEC 79 PAGE 63

REF

PAGE

1190 \* 1    /\* NOTE! 64\_REG(1) AND 64\_REG(2) ARE 64-BIT REGISTERS USED FOR DOUBLE  
1191 \* 2    /\* LENGTH INSTRUCTIONS.  
1192 \* 3    SET spp\_PRIVILEGED\_INSTRUCTION /\*INDICATE PRIVILEGED IF SPR(16,1) SET AND UC(1) SET.  
1193 \* 4    CALL GET\_AREGU(A)+16\_REG1(63,32);  
1194 \* 5    CALL SET\_AREGU((3)64\_REG1(11)31,32);  
1195 \* 6    CALL OP\_READ(64\_REG1(63,32), "DISPLACE=1")  
1196 \* 7    CALL OP\_READ(64\_REG1(31,32), "DISPLACE=0")  
1197 \* 8    IF 64\_REG(1) = 64\_REG(2)\* THEN  
1198 \* 9        ASR(2+1) I= 1 ..SET EQUAL INDICATOR.  
1199 \* 10    ELSE  
1200 \* 11        ASR(2+1) I= 0 ..INDICATE NOT EQUAL.  
1201 \* 12    ENDIF  
1202 \* 13    IF 64\_REG(1) > 64\_REG(2), THEN  
1203 \* 14        ASR(1,1) I= 1 ..INDICATE GREATER THAN OR EQUAL.  
1204 \* 15    ELSE  
1205 \* 16        ASR(1,1) I= 0 ..INDICATE LESS THAN.  
1206 \* 17    ENDIF  
1207 \* 18    RETURN  
1208 \* 19

-LMP ..LOAD BASE & MEMORY PROTECTION, PNT II F-05 4

14 DEC 79 PAGE 64

REF	PAGE	INSTRUCTION
1210	14	CALL OP_READ(132_REG11), "DISPLACE"0) ..INDIRECTION, SPR CBPR CHECKS. UC19,20) IS NOW UPDATED.
1212	14	CALL OP_READ(132_REG12), "DISPLACE"1)
1213	14	CALL GET_BREG(U(63), 32_REG13)
1214	14	32_REG13 := 32_REG13 (5,16) + U(Y)
1215	14	IF 32_REG13{0,1} "000 ADDRESS", THEN
1216	14	CALL GENERATE_SYNCHRONOUS_INTERRUPT(10P-1), CP ILLEGAL INSTRUCTION!
1217	14	..ABORT INSTRUCTION.
1218	14	ELSE IF ASR19,4)=0 "TASK MODE" AND (.NOT. ASR18){1) "NOT LOAD BASE ENABLE" .OR.
1219	14	U(S)<>7 .OR. U(A)=7), THEN
1220	14	CALL GENERATE_SYNCHRONOUS_INTERRUPT(10P-1), PRIVILEGED INSTRUCTION VIOLATION!
1221	14	11 ..ABORT INSTRUCTION.
1222	14	11 ..ABORT INSTRUCTION.
1223	14	11 ..ABORT INSTRUCTION.
1224	14	11 ..ABORT INSTRUCTION.
1225	14	11 ..ABORT INSTRUCTION.
1226	14	11 ..ABORT INSTRUCTION.
1227	14	11 ..ABORT INSTRUCTION.
1228	14	11 ..ABORT INSTRUCTION.
1229	14	11 ..ABORT INSTRUCTION.
1230	14	11 ..ABORT INSTRUCTION.
1231	14	11 ..ABORT INSTRUCTION.
1232	14	11 ..ABORT INSTRUCTION.
1233	14	11 ..ABORT INSTRUCTION.
1234	14	11 ..ABORT INSTRUCTION.
	15	CALL OP_READ(132_REG11 (17,16)) ..SET UP THE BASE REGISTER.
	15	CR10 0,160..+U(A)) := 32_REG12 (20,21) ..SET UP ASSOCIATED STORAGE PROTECTION REGISTER.
	15	CR10 0,170..+U(A)) (19,3) := U(53) ..SET UP BASE DESIGNATOR OF ASSOCIATED SIS.
	16	CR10 170..+U(A)) (15,16) := 32_REG13 (15,16) ..SET UP DISPLACEMENT OF ASSOCIATED SIS.
	16	ENDIF
	19	RETURN

\_FA (\_FAR) ..FLOATING-POINT ADD (WITH ROUND), F+F, F=06 0 & 06 4

## REF

## PAGE

```
* 1 CALL FLOATING_ADD SUB_M6 (64_REG(1), 64_REG(2))
* 2 64_REG(2) = 64_REG(1) + 64_REG(1) . 64-BIT ONE'S COMPLEMENT ADD.
1237   33 3 CALL GET_AREG (LCA), 32_REG(1) * GET INTERMEDIATE CHARACTERISTIC.
1238   3 CALL GET_AREG (LCA), 32_REG(1) * GET INTERMEDIATE CHARACTERISTIC.
1239   4 IF_OVERFLOW OCCURRED, THEN
1240   5 CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))
1241   6 ELSE
1242   7 CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))
1243   8 ENDIF
1244   9 IF (UCF2)=4 "ROUNDING OPTION", THEN
1245   10 CALL FLOATING_ROUND (64_REG(2), 32_REG(1))
1246   11 ENDIF
1247   12 CALL FLOATING_POINT_END (64_REG(2){63,32}), 32_REG(1)
1248   13 RETURN
```

NSWC  
AN/UYK-7 (CP)  
CP INSTRUCTION SET

-FAN { \_FAMR } ..FLOATING-POINT SUBTRACT (ROUND), FAN III, F=C6 1 C 06 5

REF	PAGE	
1250	*	1 CALL FLOATING_ADD_SUB_HDR (64_REG(1), 64_REG(2))
1251	*	2 64_REG(2) = 64_REG(1) - 64_REG(1) /* 64-BIT ONE'S COMPLEMENT SUBTRACT.
1252	33	3 CALL GET_AREG IU(A), 32_REG(1) /* GET INTERMEDITE CHARACTERISTIC.
1253	*	4 IF OVERFLOW OCCURRED, THEN
1254	41	5 ELSE CALL FLOATING_OVERFLOW (64_REG(2), 32_REG(1))
1255	*	6 ENDIF
1256	42	7 CALL FLOATING_NORMALIZE (64_REG(2), 32_REG(1))
1257	*	8 ENDIF
1258	*	9 IF UF2>F ROUNDING OPTION, THEN
1259	43	10 CALL FLOATING_ROUND (64_REG(2), 32_REG(1))
1260	*	11 ENDIF
1261	46	12 CALL FLOATING_POINT_END (64_REG(2)(63,32), 32_REG(1))
1262	*	13 RETURN

\_FN(\_FMR) .SFLOATING POINT MULTIPLY (ROUND), FMT II, F=0.2 E 06 6

```

REF PAGE *****
1264   * 1 SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPEC16,13 SET AND UC11 SET.
1266   * 2 CALL OP_READ(32_REG64), "DISPLACE=0) ..SET MULTIPLIER CHARACTERISTIC FROM MEMORY.
1268   * 3 CALL OP_READ(32_REG11), "DISPLACE=1) ..GET MULTIPLIED MANTISSA FROM MEMORY.
1270   * 4 CALL GET_AREG(16(A)), 32_REG63) ..GET MULTIPLICAND CHARACTERISTIC FROM A(UC11).
1272   * 5 CALL GET_AREG(16(A)+1), 32_REG62) ..GET MULTIPLICAND MANTISSA FROM A(UC11).
1274   * 6 SIGN INDICATOR I= 32_REG61(1)(31,1) ..XOR. 32_REG61(2)(31,1) ..SAVE SIGN FOR RESULT.
1276   * 7 FORCE 32_REG62, "MULTIPLIER" POSITIVE
1277   * 8 IF 32_REG611 = 0, THEN ..MULTIPLY BY ZERO.
1278   * 9 CALL PUT_AREG(16(A)), 32_REG61)
1279   * 10 CALL PUT_AREG(16(A)+1), 32_REG61)
1280   * 11 RETURN
1281   * 12
1282   * 13 32_REG611 := 32_REG61 + 32_REG64) ..FURN THE NEW CHARACTERISTIC.
1283   * 14 CALL DD_MULTIPLY ..32_REG62) X 32_REG61) WITH 64-BIT RESULT PUT IN REPEAT_ACCUMULATOR 6
1284   * 15 SHIFT REPEAT_ACCUMULATOR AND 32_REG611 PAIR LEFT LOGICAL ONE ..CORRECT FOR FRACTIONAL MULTIPLY.
1285   * 16 IF REPEAT_ACCUMULATOR(30,1) = 0, THEN ..NORMALIZE.
1286   * 17 SHIFT REPEAT_ACCUMULATOR AND 32_REG611 PAIR LEFT LOGICAL ONE
1287   * 18 32_REG63 := 32_REG63 - 1 ..UPDATE CHARACTERISTIC.
1288   * 19 ENDIF
1289   * 20 IF UCF21 = 6 "ROUNDING OPTION", THEN
1290   * 21 CALL FLOATING_ROUND (REPEAT_ACCUMULATOR, 32_REG63))
1291   * 22 ENDIF
1292   * 23 IF SIGN_INDICATOR "NEGATIVE", THEN ..DO SIGN CORRECTION.
1293   * 24 REPEAT_ACCUMULATOR := .NOT. REPEAT_ACCUMULATOR
1294   * 25 ENDIF
1295   * 26 CALL FLOATING_POINT_END (REPEAT_ACCUMULATOR, 32_REG63)
1296   * 27 RETURN
1297   * 28
1298   * 29

```

-FC1\_FDR1 ..FLOATING-POINT DIVIDE (WITH ROUND), FNT II, F-06 3 E 06 7

REF

```

PAGE *      1 SET SPR_PRIVILEGED_INSTRUCTION =1; SET AND UIJ SET.
1300   * 1 CALL DP_READ (32_REG10), "DISPLACE0) .-GET DIVISION CHARACTERISTIC FROM MEMORY.
1302   * 2 CALL DP_READ (32_REG10), "DISPLACE1) .-GET DIVISION MANTISSA FROM MEMORY.
1304   * 3 CALL DP_READ (32_REG10), "DISPLACE2) .-GET DIVISION MANTISSA FROM MEMORY.
1306   * 4 CALL GET_AREG (U1A), 32_REG(3) .-GET DIVISION CHARACTERISTIC FROM AREG(A).
1308   * 5 CALL GET_AREG(U1A), 64_REG(1)(63,32) .-GET DIVISION MANTISSA FROM AREG(A)+1.
1310   * 6 SIGN_IND := 64_REG(1)(63,1) .-XOR 32_REG(1)(31,1) .-SIGN INDICATOR.
1311   * 7 FORCE 64_REG(1) "DIVISOR" AND 32_REG(1) "DIVISOR" POSITIVE
1312   * 8 IF 64_REG(1)(63,32) "DIVISOR" 0, THEN
1313   * 9 32_REG(3) := 0 ..FORCE DIVISION CHARACTERISTIC TO ZERO.
1314   * 10 ENDIF
1315   * 11 IF 32_REG(11) = 0 "DIVISOR ZERO", THEN
1316   * 12 CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P=1"), FLOATING POINT OVERFLOW)
1317   * 13 ENDIF
1318   * 14 32_REG(13) := 32_REG(12) ..COMPUTE INTERMEDIATE CHARACTERISTIC.
1320   * 15 CALL DIVIDE_COMPARE (64_REG(11), 32_REG(11))
1321   * 16 COUNT := 31
1322   * 17 BG WHILE COUNT > 0
1323   * 18 64_REG(11) := 64_REG(11) *LL 1
1324   * 19 CALL DIVIDE_COMPARE (64_REG(11), 32_REG(11))
1325   * 20 COUNT := COUNT -1
1326   * 21 ENDOD ..END WHILE.
1327   * 22 64_REG := 64_REG SHIFTED LEFT CIRCULAR 32 BITS
1328   * 23 ..64_REG CAN NOW SHARE TERMINATION LOGIC WITH OTHER FLOATING POINT ROUTINES.
1329   * 24 ..NOTES: 64_REG(63,32) NOW HAS THE QUOTIENT AND 64_REG(31,32) HAS THE REMAINDER.
1330   * 25 IF UCF2>3 "NO ROUNDING", THEN
1331   * 26 IF 64_REG(1)(63,1)=1 "OVERFLOW OCCURRED", THEN
1332   * 27 CALL FLOATING_OVERFLOW (64_REG(11), 32_REG(13)) ..UPDATE MANTISSA AND CHARACTERISTIC.
1333   * 28 ENDIF
1334   * 29 ELSE .."ROUNDING".
1335   * 30 IF 64_REG(1)(63,1)=1 "OVERFLOW OCCURRED", THEN
1336   * 31 64_REG(1)(32,1) := 64_REG(1)(32,1) ..SAVE ROUND INDICATOR.
1337   * 32 CALL FLOATING_OVERFLOW (64_REG(11), 32_REG(13)) ..UPDATE MANTISSA AND CHARACTERISTIC.
1338   * 33 ENDIF
1339   * 34 CALL ROUND_UP (64_REG(11)(63,32), 32_REG(13))
1340   * 35 ENDIF
1341   * 36 CALL ROUND_UP (64_REG(11)(63,32), 32_REG(13))
1342   * 37 ENDIF
1343   * 38 CALL ROUND_UP (64_REG(11)(63,32)) ..REMAINDER MPLUS 1/2 THE DIVISOR.
1344   * 39 ENDIF
1345   * 40 ENDIF
1346   * 41 IF SIGN_IND SET, THEN ..CORRECT SIGN.
1347   * 42 IF 64_REG(1)(63,32) == .NOT1. 64_REG(1)(63,32)
1348   * 43 ENDIF
1349   * 44 ENDIF
1350   * 45 CALL FLOATINT_POINT_EM8 (64_REG(11)(63,32), 32_REG(13)) ..FINISH THE INSTRUCTION.
1351   * 46 RETURN
1352   * 47
1353   * 48
1354   * 49
1355   * 50
1356   * 51
1357   * 52
1358   * 53
1359   * 54
1360   * 55
1361   * 56
1362   * 57
1363   * 58
1364   * 59
1365   * 60
1366   * 61
1367   * 62
1368   * 63
1369   * 64
1370   * 65
1371   * 66
1372   * 67
1373   * 68
1374   * 69
1375   * 70
1376   * 71
1377   * 72
1378   * 73
1379   * 74
1380   * 75
1381   * 76
1382   * 77
1383   * 78
1384   * 79
1385   * 80
1386   * 81
1387   * 82
1388   * 83
1389   * 84
1390   * 85
1391   * 86
1392   * 87
1393   * 88
1394   * 89
1395   * 90
1396   * 91
1397   * 92
1398   * 93
1399   * 94
1400   * 95
1401   * 96
1402   * 97
1403   * 98
1404   * 99
1405   * 100
1406   * 101
1407   * 102
1408   * 103
1409   * 104
1410   * 105
1411   * 106
1412   * 107
1413   * 108
1414   * 109
1415   * 110
1416   * 111
1417   * 112
1418   * 113
1419   * 114
1420   * 115
1421   * 116
1422   * 117
1423   * 118
1424   * 119
1425   * 120
1426   * 121
1427   * 122
1428   * 123
1429   * 124
1430   * 125
1431   * 126
1432   * 127
1433   * 128
1434   * 129
1435   * 130
1436   * 131
1437   * 132
1438   * 133
1439   * 134
1440   * 135
1441   * 136
1442   * 137
1443   * 138
1444   * 139
1445   * 140
1446   * 141
1447   * 142
1448   * 143
1449   * 144
1450   * 145
1451   * 146
1452   * 147
1453   * 148
1454   * 149
1455   * 150
1456   * 151
1457   * 152
1458   * 153
1459   * 154
1460   * 155
1461   * 156
1462   * 157
1463   * 158
1464   * 159
1465   * 160
1466   * 161
1467   * 162
1468   * 163
1469   * 164
1470   * 165
1471   * 166
1472   * 167
1473   * 168
1474   * 169
1475   * 170
1476   * 171
1477   * 172
1478   * 173
1479   * 174
1480   * 175
1481   * 176
1482   * 177
1483   * 178
1484   * 179
1485   * 180
1486   * 181
1487   * 182
1488   * 183
1489   * 184
1490   * 185
1491   * 186
1492   * 187
1493   * 188
1494   * 189
1495   * 190
1496   * 191
1497   * 192
1498   * 193
1499   * 194
1500   * 195
1501   * 196
1502   * 197
1503   * 198
1504   * 199
1505   * 200
1506   * 201
1507   * 202
1508   * 203
1509   * 204
1510   * 205
1511   * 206
1512   * 207
1513   * 208
1514   * 209
1515   * 210
1516   * 211
1517   * 212
1518   * 213
1519   * 214
1520   * 215
1521   * 216
1522   * 217
1523   * 218
1524   * 219
1525   * 220
1526   * 221
1527   * 222
1528   * 223
1529   * 224
1530   * 225
1531   * 226
1532   * 227
1533   * 228
1534   * 229
1535   * 230
1536   * 231
1537   * 232
1538   * 233
1539   * 234
1540   * 235
1541   * 236
1542   * 237
1543   * 238
1544   * 239
1545   * 240
1546   * 241
1547   * 242
1548   * 243
1549   * 244
1550   * 245
1551   * 246
1552   * 247
1553   * 248
1554   * 249
1555   * 250
1556   * 251
1557   * 252
1558   * 253
1559   * 254
1560   * 255
1561   * 256
1562   * 257
1563   * 258
1564   * 259
1565   * 260
1566   * 261
1567   * 262
1568   * 263
1569   * 264
1570   * 265
1571   * 266
1572   * 267
1573   * 268
1574   * 269
1575   * 270
1576   * 271
1577   * 272
1578   * 273
1579   * 274
1580   * 275
1581   * 276
1582   * 277
1583   * 278
1584   * 279
1585   * 280
1586   * 281
1587   * 282
1588   * 283
1589   * 284
1590   * 285
1591   * 286
1592   * 287
1593   * 288
1594   * 289
1595   * 290
1596   * 291
1597   * 292
1598   * 293
1599   * 294
1600   * 295
1601   * 296
1602   * 297
1603   * 298
1604   * 299
1605   * 300
1606   * 301
1607   * 302
1608   * 303
1609   * 304
1610   * 305
1611   * 306
1612   * 307
1613   * 308
1614   * 309
1615   * 310
1616   * 311
1617   * 312
1618   * 313
1619   * 314
1620   * 315
1621   * 316
1622   * 317
1623   * 318
1624   * 319
1625   * 320
1626   * 321
1627   * 322
1628   * 323
1629   * 324
1630   * 325
1631   * 326
1632   * 327
1633   * 328
1634   * 329
1635   * 330
1636   * 331
1637   * 332
1638   * 333
1639   * 334
1640   * 335
1641   * 336
1642   * 337
1643   * 338
1644   * 339
1645   * 340
1646   * 341
1647   * 342
1648   * 343
1649   * 344
1650   * 345
1651   * 346
1652   * 347
1653   * 348
1654   * 349
1655   * 350
1656   * 351
1657   * 352
1658   * 353
1659   * 354
1660   * 355
1661   * 356
1662   * 357
1663   * 358
1664   * 359
1665   * 360
1666   * 361
1667   * 362
1668   * 363
1669   * 364
1670   * 365
1671   * 366
1672   * 367
1673   * 368
1674   * 369
1675   * 370
1676   * 371
1677   * 372
1678   * 373
1679   * 374
1680   * 375
1681   * 376
1682   * 377
1683   * 378
1684   * 379
1685   * 380
1686   * 381
1687   * 382
1688   * 383
1689   * 384
1690   * 385
1691   * 386
1692   * 387
1693   * 388
1694   * 389
1695   * 390
1696   * 391
1697   * 392
1698   * 393
1699   * 394
1700   * 395
1701   * 396
1702   * 397
1703   * 398
1704   * 399
1705   * 400
1706   * 401
1707   * 402
1708   * 403
1709   * 404
1710   * 405
1711   * 406
1712   * 407
1713   * 408
1714   * 409
1715   * 410
1716   * 411
1717   * 412
1718   * 413
1719   * 414
1720   * 415
1721   * 416
1722   * 417
1723   * 418
1724   * 419
1725   * 420
1726   * 421
1727   * 422
1728   * 423
1729   * 424
1730   * 425
1731   * 426
1732   * 427
1733   * 428
1734   * 429
1735   * 430
1736   * 431
1737   * 432
1738   * 433
1739   * 434
1740   * 435
1741   * 436
1742   * 437
1743   * 438
1744   * 439
1745   * 440
1746   * 441
1747   * 442
1748   * 443
1749   * 444
1750   * 445
1751   * 446
1752   * 447
1753   * 448
1754   * 449
1755   * 450
1756   * 451
1757   * 452
1758   * 453
1759   * 454
1760   * 455
1761   * 456
1762   * 457
1763   * 458
1764   * 459
1765   * 460
1766   * 461
1767   * 462
1768   * 463
1769   * 464
1770   * 465
1771   * 466
1772   * 467
1773   * 468
1774   * 469
1775   * 470
1776   * 471
1777   * 472
1778   * 473
1779   * 474
1780   * 475
1781   * 476
1782   * 477
1783   * 478
1784   * 479
1785   * 480
1786   * 481
1787   * 482
1788   * 483
1789   * 484
1790   * 485
1791   * 486
1792   * 487
1793   * 488
1794   * 489
1795   * 490
1796   * 491
1797   * 492
1798   * 493
1799   * 494
1800   * 495
1801   * 496
1802   * 497
1803   * 498
1804   * 499
1805   * 500
1806   * 501
1807   * 502
1808   * 503
1809   * 504
1810   * 505
1811   * 506
1812   * 507
1813   * 508
1814   * 509
1815   * 510
1816   * 511
1817   * 512
1818   * 513
1819   * 514
1820   * 515
1821   * 516
1822   * 517
1823   * 518
1824   * 519
1825   * 520
1826   * 521
1827   * 522
1828   * 523
1829   * 524
1830   * 525
1831   * 526
1832   * 527
1833   * 528
1834   * 529
1835   * 530
1836   * 531
1837   * 532
1838   * 533
1839   * 534
1840   * 535
1841   * 536
1842   * 537
1843   * 538
1844   * 539
1845   * 540
1846   * 541
1847   * 542
1848   * 543
1849   * 544
1850   * 545
1851   * 546
1852   * 547
1853   * 548
1854   * 549
1855   * 550
1856   * 551
1857   * 552
1858   * 553
1859   * 554
1860   * 555
1861   * 556
1862   * 557
1863   * 558
1864   * 559
1865   * 560
1866   * 561
1867   * 562
1868   * 563
1869   * 564
1870   * 565
1871   * 566
1872   * 567
1873   * 568
1874   * 569
1875   * 570
1876   * 571
1877   * 572
1878   * 573
1879   * 574
1880   * 575
1881   * 576
1882   * 577
1883   * 578
1884   * 579
1885   * 580
1886   * 581
1887   * 582
1888   * 583
1889   * 584
1890   * 585
1891   * 586
1892   * 587
1893   * 588
1894   * 589
1895   * 590
1896   * 591
1897   * 592
1898   * 593
1899   * 594
1900   * 595
1901   * 596
1902   * 597
1903   * 598
1904   * 599
1905   * 600
1906   * 601
1907   * 602
1908   * 603
1909   * 604
1910   * 605
1911   * 606
1912   * 607
1913   * 608
1914   * 609
1915   * 610
1916   * 611
1917   * 612
1918   * 613
1919   * 614
1920   * 615
1921   * 616
1922   * 617
1923   * 618
1924   * 619
1925   * 620
1926   * 621
1927   * 622
1928   * 623
1929   * 624
1930   * 625
1931   * 626
1932   * 627
1933   * 628
1934   * 629
1935   * 630
1936   * 631
1937   * 632
1938   * 633
1939   * 634
1940   * 635
1941   * 636
1942   * 637
1943   * 638
1944   * 639
1945   * 640
1946   * 641
1947   * 642
1948   * 643
1949   * 644
1950   * 645
1951   * 646
1952   * 647
1953   * 648
1954   * 649
1955   * 650
1956   * 651
1957   * 652
1958   * 653
1959   * 654
1960   * 655
1961   * 656
1962   * 657
1963   * 658
1964   * 659
1965   * 660
1966   * 661
1967   * 662
1968   * 663
1969   * 664
1970   * 665
1971   * 666
1972   * 667
1973   * 668
1974   * 669
1975   * 670
1976   * 671
1977   * 672
1978   * 673
1979   * 674
1980   * 675
1981   * 676
1982   * 677
1983   * 678
1984   * 679
1985   * 680
1986   * 681
1987   * 682
1988   * 683
1989   * 684
1990   * 685
1991   * 686
1992   * 687
1993   * 688
1994   * 689
1995   * 690
1996   * 691
1997   * 692
1998   * 693
1999   * 694
2000   * 695
2001   * 696
2002   * 697
2003   * 698
2004   * 699
2005   * 700
2006   * 701
2007   * 702
2008   * 703
2009   * 704
2010   * 705
2011   * 706
2012   * 707
2013   * 708
2014   * 709
2015   * 710
2016   * 711
2017   * 712
2018   * 713
2019   * 714
2020   * 715
2021   * 716
2022   * 717
2023   * 718
2024   * 719
2025   * 720
2026   * 721
2027   * 722
2028   * 723
2029   * 724
2030   * 725
2031   * 726
2032   * 727
2033   * 728
2034   * 729
2035   * 730
2036   * 731
2037   * 732
2038   * 733
2039   * 734
2040   * 735
2041   * 736
2042   * 737
2043   * 738
2044   * 739
2045   * 740
2046   * 741
2047   * 742
2048   * 743
2049   * 744
2050   * 745
2051   * 746
2052   * 747
2053   * 748
2054   * 749
2055   * 750
2056   * 751
2057   * 752
2058   * 753
2059   * 754
2060   * 755
2061   * 756
2062   * 757
2063   * 758
2064   * 759
2065   * 760
2066   * 761
2067   * 762
2068   * 763
2069   * 764
2070   * 765
2071   * 766
2072   * 767
2073   * 768
2074   * 769
2075   * 770
2076   * 771
2077   * 772
2078   * 773
2079   * 774
2080   * 775
2081   * 776
2082   * 777
2083   * 778
2084   * 779
2085   * 780
2086   * 781
2087   * 782
2088   * 783
2089   * 784
2090   * 785
2091   * 786
2092   * 787
2093   * 788
2094   * 789
2095   * 790
2096   * 791
2097   * 792
2098   * 793
2099   * 794
2100   * 795
2101   * 796
2102   * 797
2103   * 798
2104   * 799
2105   * 800
2106   * 801
2107   * 802
2108   * 803
2109   * 804
2110   * 805
2111   * 806
2112   * 807
2113   * 808
2114   * 809
2115   * 810
2116   * 811
2117   * 812
2118   * 813
2119   * 814
2120   * 815
2121   * 816
2122   * 817
2123   * 818
2124   * 819
2125   * 820
2126   * 821
2127   * 822
2128   * 823
2129   * 824
2130   * 825
2131   * 826
2132   * 827
2133   * 828
2134   * 829
2135   * 830
2136   * 831
2137   * 832
2138   * 833
2139   * 834
2140   * 835
2141   * 836
2142   * 837
2143   * 838
2144   * 839
2145   * 840
2146   * 841
2147   * 842
2148   * 843
2149   * 844
2150   * 845
2151   * 846
2152   * 847
2153   * 848
2154   * 849
2155   * 850
2156   * 851
2157   * 852
2158   * 853
2159   * 854
2160   * 855
2161   * 856
2162   * 857
2163   * 858
2164   * 859
2165   * 860
2166   * 861
2167   * 862
2168   * 863
2169   * 864
2170   * 865
2171   * 866
2172   * 867
2173   * 868
2174   * 869
2175   * 870
2176   * 871
2177   * 872
2178   * 873
2179   * 874
2180   * 875
2181   * 876
2182   * 877
2183   * 878
2184   * 879
2185   * 880
2186   * 881
2187   * 882
2188   * 883
2189   * 884
2190   * 885
2191   * 886
2192   * 887
2193   * 888
2194   * 889
2195   * 890
2196   * 891
2197   * 892
2198   * 893
2199   * 894
2200   * 895
2201   * 896
2202   * 897
2203   * 898
2204   * 899
2205   * 900
2206   * 901
2207   * 902
2208   * 903
2209   * 904
2210   * 905
2211   * 906
2212   * 907
2213   * 908
2214   * 909
2215   * 910
2216   * 911
2217   * 912
2218   * 913
2219   * 914
2220   * 915
2221   * 916
2222   * 917
2223   * 918
2224   * 919
2225   * 920
2226   * 921
2227   * 922
2228   * 923
2229   * 924
2230   * 925
2231   * 926
2232   * 927
2233   * 928
2234   * 929
2235   * 930
2236   * 931
2237   * 932
2238   * 933
2239   * 934
2240   * 935
2241   * 936
2242   * 937
2243   * 938
2244   * 939
2245   * 940
2246   * 941
2247   * 942
2248   * 943
2249   * 944
2250   * 945
2251   * 946
2252   * 947
2253   * 948
2254   * 949
2255   * 950
2256   * 951
2257   * 952
2258   * 953
2259   * 954
2260   * 955
2261   * 956
2262   * 957
2263   * 958
2264   * 959
2265   * 960
2266   * 961
2267   * 962
2268   * 963
2269   * 964
2270   * 965
2271   * 966
2272   * 967
2273   * 968
2274   * 969
2275   * 970
2276   * 971
2277   * 972
2278   * 973
2279   * 974
2280   * 975
2281   * 976
2282   * 977
2283   * 978
2284   * 979
2285   * 980
2286   * 981
2287   * 982
2288   * 983
2289   * 984
2290   * 985
2291   * 986
2292   * 987
2293   * 988
2294   * 9
```

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

16 DEC 79 PAGE 69

-X5 ..ENTER EXECUTIVE STATE, FM1 II F=07 O A=0

REF PAGE \*\*\*\*\*  
1360 \* 1 IE UC1) "INDICTION", THEN  
1361 \* 2 CALL IA\_SEQUENCE(DUMMY PARAMETERS) --OP\_READ NOT USED SINCE OPERAND IS NOT FETCHED.  
1362 \* 3 ENDIF  
1363 \* 4 CALL GET\_BREG (UC(B),32,REG(1))  
1364 \* 5 CODE I\_16SY) + 32\_PEGC1(15,16) .16-BIT ISC TO BE STORED UPON INTERRUPT.  
1365 \* 6 CALL GENERATE\_SYNCHRONOUS\_INTERRUPT (\*P="G, ENTER EXECUTIVE STATE, CODE1)  
1366 \* 7 SEIUEJ  
1369 \*

IPI .. INTERPROCESSOR INTERRUPT, FMT II F=07 G A=1

REF

PAGE

```
*      * 1   IF ASR(19,4)=0 "TASK MODE", THEN
1371    * 2     CALL GENERATE_SYNCHRONOUS_INTERRUPT ("P->L", PRIVILEGED INSTRUCTION VIOLATION)
1372    * 3     ..ABORT INSTRUCTION.
1373    * 4
1374    * 5   ELSE
1375    * 6     IF U(1) "INDIRECTION", THEN
1376    * 7       CALL TA_SEQUENCE (OUNTY PARAMETERS) ..EP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1377    * 8
1378    * 9     CALL GET_REG (U(18), 32_REG(1))
1379    * 10    32_REG(1) := U(SY) + 32_REG(1) --16-BIT QUANTITY.
1380    * 11    IF 32_REG(1)<15,1>, THEN
1381    * 12      32_REG(1)(CPB,1) := 0 ..PREVENT INTERRUPTION OF SELF.
1382    * 13
1383    * 14
1384    * 15
1385    * 16
1386    * 17
1387    * 18
1388    * 19
1389    * 20
1390    * 21
1391    * 22
1392    * 23
1393    * 24
```

-AEI (C -PEI) ..ALIGN & PREVENT ENABLE INTERRUPTS, FMT III F=07 1 E 2

```

REF PAGE *****
PAGE * 1 IF ASR(C9,4)=0 "TASK MODE", THEN
      * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT (-P-1), PRIVILEGED INSTRUCTION VIOLATION
1395 11 * 3 ..ABORT INSTRUCTION.
1396 * 4
1398 * 5 IF UC(1) "INDIRECTION", THEN
1400 * 6 CALL IA_SEQUENCE (DUMMY PARAMETERS) ..OP_READ NOT USED SINCE OPERAND IS NOT FETCHED.
1401 * 7
1402 * 8 ENDIF
1403 * 9 CALL GET_BREG(CB), 32_REG(M) ..COMPUTE LOW 16-BITS OF O_BUS.
1404 * 10 32_REG(1) := UC(Y) + 32_REG(1) ..FUNCTION CODE AND VALUE TO O_BUS.
1405 * 11 INITIATE AN IUC REQUEST ON IUC(U(A),A,3)
1406 * 12 C_BUS := IUC(F1..LL-263 + 32REG(1){15,16}) ..FUNCTION CODE AND VALUE TO O_BUS.
1407 * 13 SEND O_BUS TO IUC(U(A),A,3)
1408 * 14 IF REQUEST NOT HONORED -OR- O_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN
1409 * 15 CALL GENERATE_SYNCHRONOUS_INTERRUPT (-P-1), CP-IUC COMMAND RESUME, U(A);
1410 * 16 ..ABORT INSTRUCTION.
1411 * 17 ENDIF
1412 * 18 RETURN
1413 * 19
1414 * 20
1415 * 21
1416 * 22

```

NSWC  
AN/UVK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 72

\_LIM ==>JAD, ENABLE IOC MONITOR CLOCK, FMT II F=07 3

REF PAGE \*\*\*\*\*  
1418 \* 1 IF ASFC19,4=0 "MASK #ODE", THEN  
1419 \* 2 CALL GENERATE\_SYNCHRONOUS\_INTERRUPT {"P"}, PRIVILEGED INSTRUCTION VIOLATION.  
1420 \* 3 ..ABORT THE INSTRUCTION.  
1422 \* 4 ELSE  
1423 \* 5 IF U(I) "INDIRECTION", THEN ..OP\_PREAD NOT USED SINCE OPERAND IS NOT FETCHED.  
1424 \* 6 CALL TA\_SEQUENCE (OURRY PARAMETERS) ..OP\_PREAD NOT USED SINCE OPERAND IS NOT FETCHED.  
1426 \* 7 ENDIF  
1427 \* 8 CALL GET\_BREG(U), 32\_REGC11  
1428 \* 9 32\_REGC11 := U(SY) + 32\_REGC11 ..COMPUTE LGW 16-BITS OF O\_BUS.  
1429 \* 10 INITIATE AN IOC REQUEST ON IOC(U(I)).A.3  
1430 \* 11 O\_BUS := (U(F).LL+26) + (U(F2).LL+20) +32\_REGC11(K5,16) ..FUNCTION CODE AND VALUE TO O\_BUS.  
1~32 \* 12 SEND O\_BUS TO IOC(U(A)).A.3  
1433 \* 13 IF REQUEST NOT MONORED ..OR.. O\_BUS NOT RECEIVED WITHIN ACCEPTABLE TIME FRAME, THEN  
1435 \* 14 CALL GENERATE\_SYNCHRONOUS\_INTERRUPT ("P=1, CP-ICC COMMAND RESUME, U(A))  
1436 \* 15 ..ABORT INSTRUCTION.  
1437 \* 16 ENDIF  
1438 \* 17 ENDIF  
1439 \* 18 RETURN  
\*\*\*\*\*

\_10 ..INITIATE I/C. FMT II F=07 ..

REF  
PAGE

```

1441   * 1 IF ASF(15,4)=0 "TASK MODE", THEN
1442   * 2 CALL GENERATE_SYNCRONOUS_INTERRUPT ("P=1"), PRIVILEGED INSTRUCTION VIOLATION)
1443   * 3 ..ABORT THE INSINUATION.
1444
1445   * 4 ELSE
1446   * 5   IF U(1) "INPECT ADDRESSING", THEN
1447   * 6     CALL IA_SEQUENCE (S_DESIGNATOR, Y, DUMMY PARAMETERS) --DO CASCADING AS REQUIRED.
1448   * 7     ELSE --DEFINE MCAPIAL BASE (S) REGISTER DESIGNATOR & DISPLACEMENT.
1449   * 8       S_DESIGNATOR := U(5) --DEFINE BASE (S) REGISTER SELECTOR.
1450   * 9       CALL GET_REG (U(6), 32, REG1)
1451   * 10      Y := U(") + 32_REG1)(15) ..COMPUTE DISPLACEMENT.
1452
1453   * 11 ENDIF
1454   * 12 CALL ADD_S (S_DESIGNATOR, Y) ..COMPLETE ABSOLUTE ADDRESS.
1455   * 13 INITIATE AN I/C REQUEST ON IOCT(UA).A.3).
1456   * 14   C_BUS := U(F).LL.26 + U(F2).LL.29 + Y(17) ..FUNCTION DESIGNATORS & ABSOLUTE ADDRESS.
1457
1458   * 15   SEND C BUS TO ICC(UA).A.31
1459   * 16   IF .NOT. (I/C REQUEST MONOSED OR Q_BUS RECEIVED) WITHIN ACCEPTABLE TIME FRAME, THEN
1460   * 17     CALL GENERATE_SYNCRONOUS_INTERRUPT (P=1, ICC_COMMAND_RESUME, U(A).A.31)
1461
1462   * 18     ..ABORT THIS INSTRUCTION.
1463
1464   * 19 ENDIF
1465   * 20
1466   * 21 RETURN

```

NSNC

ANS/CRK-7 (CP)  
CF INSTRUCTION SET

14 DEC 79 PAGE 74

\_IS . . INTERRUPT RETURN, FMT II f=07 5

REF

PAGE \*

\*\*\*\*\*

```
1468 * 1 IF ASPL(9,4)=0 /*TASK MODE", THEN
1469 * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT (*P=1, PRIVILEGED INSTRUCTION VIOLATION)
1470 * 3 . .ABORT THIS INSTRUCTION.
1471 * 4 ELSE CURRENT_INTERRUPT_LEVEL (STATE) ..AS INDICATED BY ACTIVE STATUS REGISTER.
1472 * 5 CLASS 1 = CMR10*35*(4*CLASS) /*RESTORE ACTIVE STATUS REGISTER.
1473 * 6 ASR := CMR10*35*(4*CLASS) /*RESTORE ACTIVE STATUS REGISTER.
1474 * 7 P := CMR10*137*(4*CLASS) .RESTORE PROGRAM COUNTER.
1475 * 8 ENDIF
1476 * 9 RETURN
1477 *
1478 *
```

\*\*\*\*\*

RP ..REPEAT, FMT II F-07 6

REF PAGE	1	.. THE INSTRUCTION SEQUENCE WILL CHECK FOR REPEATABILITY OF THE NEXT .. INSTRUCTION, AND IF REPEATABLE THE REPEAT INDICATOR WILL BE SET. .. SHOULD THE NEXT INSTRUCTION BE FOUND NOT TO BE REPEATABLE (AS PER .. THE REPERTOIRE CARD), THAT INSTRUCTION WILL BE ABORTED AND REPEAT .. MODE TERMINATED.
1480	2	IF U(C1) "INDIRECT ADDRESSING", THEN CALL IA_SEQUENCE (DUMMY PARAMETERS) ..DO CASCADING AS REQUIRED.
1481	3	ENDIF
1482	4	IF B(7) = 0, THEN ..SKIP NEXT INSTRUCTION.
1483	5	P(D) := P(D) + 1 ..INCREMENT PROGRAM COUNTER TO THE ADDRESS JUST AFTER THAT OF THE REPEATED INSTRUCTION.
1484	6	ELSE ..EXECUTE NEXT INSTRUCTION B(7) TIMES OR UNTIL THE CONDITION INDICATED BY U(A). SET REPEAT_PENDING INDICATOR
1485	7	SAVE U(A) & U(B) IN REPEAT_AB FOR REPEAT_SEQUENCE USAGE
1486	8	SAVE U(SY) IN REPEAT_SY
1487	9	ENDIF
1488	10	RETURN
1489	11	*
1490	12	*
1491	13	*
1492	14	*
1493	15	*
1494	16	*
1495	17	*
1496	18	*
1497	19	*

NSC

AB/UTK-7 (CF)  
CP INSTRUCTION SET

-LA . . LOAD ACCUMULATOR WITH MEMORY CONTENTS, FMT I F=10

REF

PAGE

1499	14	1	CALL GPREAD (REPEAT_ACCUMULATOR, "DISPLACE01 ..FETCH OPERAND AS PER K DESIGNATOR.
1501	36	2	CALL PUT_AREG (LFA), REPEAT_ACCUMULATOR)
1502	3	3	EE11E8
*****			

14 DEC 79 PAGE 76

NSWC AN/UYK-7 [CPI]  
CP INSTRUCTION SET

14 DEC 79 PAGE 77

\_LKB ..LOAD ACCUMULATOR & INDEX B, FMT 1, Fall

REF PAGE \*\*\*\*\*  
1504 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION\_INDICATOR ..LKB IS A SPECIAL CASE FOR SPR(16,1)=1 & INDIRECT  
\* ADDRESSING.  
1506 \* 2 CALL OP\_READ (REF(EA)-ACCUMULATOR, "DISPLACE=0")  
1507 \* 3 CALL PUT\_AREG (LUA), REPEAT\_ACCUMULATOR)  
1508 \* 4 CALL GET\_BREG (LUB), 32\_REG(1)  
1509 \* 5 32\_REG(1) := 32\_P6(1){15} + 1 ..INCREMENT INDEX (B) REGISTER VALUE.  
1510 \* 6 CALL PUT\_BREG (LUB), 32\_REG(1)  
1511 \* 7 RETURN

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 78

\_LDIF ==LOAD DIFFERENCE, FMT I, F=12

REF PAGE \*\*\*\*\*

1513	*	1	CALL OP_PEAAC (REPEAT_ACCUMULATOR, "DISPLACE" 0)
1514	*	2	CALL GET_AREG (UCA), 32-PREG(2)
1515	*	3	REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR - 32_PEG(2) ..ONES COMPLEMENT SUBTRACT.
1517	*	4	UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)
1516	*	5	CALL PUT_AREG (UCA)+1, REPEAT_ACCUMULATOR)
1519	*	6	RETDRA

NSWC AN/UYK-7 (UCP)  
CP INSTRUCTION SET

16 DEC 79 PAGE 79

\_ANA ==SUBTRACT A, FMT I, F=13

REF  
PAGE

1521	14	*	1	CALL UP_READ (32_REG, "DISPLACE"=0)
1522	33	*	2	CALL GET_REG (U(A), REPEAT_ACCUMULATOR)
1523	*	*	3	REPEAT_ACCUMULATOR = REPEAT_ACCUMULATOR - 32_REG(2) ==ONES COMPLEMENT SUBTRACT.
1525	*	*	4	UPDATE FIXED PCINT CVERFLOW BIT IN ASR (BIT 3)
1526	36	*	5	CALL PUT_REG (U(A), REPEAT_ACCUMULATOR)
1527	*	*	6	RETURN

NSWC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

\_AA ..ADD A, FRT I, F=14

14 DEC 79 PAGE 80

REF PAGE \*\*\*\*\*

1529	16	*	1	CALL OP_READ (32_REG(2), "DISPLACE" 0)
1530	33	*	2	CALL GET_AREG (U(A)), REPEAT_ACCUMULATOR)
1531	*	3	REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) ..ONES COMPLEMENT ADDITION.	
1533	*	4	UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)	
1534	36	*	5	CALL PUT_AREG (U(A)), REPEAT_ACCUMULATOR)
1535	*	6	*	RETURNS
		*	*	*****

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 SEC 79 PAGE 81

LSUM ..LOAD SUM, FM1 I, F=15

REF PAGE \*\*\*\*\*

```
1537    14   * 1 CALL CP_READ(32_REG(2), "DISPLACED")
1538    33   * 2 CALL GET_AREG([A], REPEAT_ACCUMULATOR)
1539    * 3 REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR + 32_REG(2) **ONES COMPLEMENT ADDITION**
1540    * 4 UPDATE FIXED POINT OVERFLOW BIT IN ASR (BIT 3)
1541    36   * 5 CALL PUT_AREG([A]+1, REPEAT_ACCUMULATOR)
1542    * 6 RETURN
1543    * 7 *****
```

\_LNA --LOAD NEGATIVE, FMT I, F=16

REF PAGE \*\*\*\*\*

1545	14	*	1	CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE" 0)
1546	*	2	*	REPEAT_ACCUMULATOR :- REPEAT_ACCUMULATOR ..ONES COMPLEMENT.
1547	36	*	3	CALL PUT_AREC (U(A), REPEAT_ACCUMULATOR)
1548	*	4	*	RETURN
		*		*****

MSMC

AN/UVK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 63

\_LN ..LOAD MAGNITUDE, FRT 1, FR=17

REF PAGE \*\*\*\*\*

1550	14	1	CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE" 0)
1551	*	2	IE REPEAT_ACCUMULATOR < 0, THEN
1552	*	3	REPEAT_ACCUMULATOR := - REPEAT_ACCUMULATOR == ONES COMPLEMENT.
1553	*	4	ENDIF
1554	36	5	CALL PUT_AREG(UCA), REPEAT_ACCUMULATOR)
1555	*	6	RETIRE

AN/UVK-7 'CPI'  
CP INSTRUCTION SET

\_L8 ..LOAD B, FR1 I F=20

REF PAGE

*	*	1	* Y -> R(C1A)(15)
1557	14	2	CALL OP_READ(32_REG(1)), "DISPLACE"=01
1558	34	3	CALL GET_AREG(U(A), 32_REG(2))
1559	*	4	32_REG(1)(19,3) = 32_REG(1)(19,3)
1560	*	5	CALL PUT_AREG(U(A), 32_REG(1))
1561	37	*	RETURNS
1562	*	6	

NSMC            AN/UVK-7 (CP)  
                  CP INSTRUCTION SET

-38 . .ADD B, INT 1 F = 21

REF

PAGE

1564        1        :: B(LU(A))(15) \* Y -> B(LU(A))(15)  
1565        2        CALL CP\_READ(12, REG(1), "DISPLACE=0")  
1566        3        CALL GET\_BREG(U(A), 32\_REG(2))  
1567        4        32\_REG(3) := REPEAT\_ACCUMULATOR(15)  
1568        5        32\_REG(3) := 22\_REG(3) + 32\_REG(1) ..ONES COMPLEMENT ADDITION.  
1569        6        REPEAT\_ACCUMULATOR(15) :: 32\_REG(3)  
1570        7        CALL PUT\_BREG(U(A), REPEAT\_ACCUMULATOR)  
1571        8        RETBN

NSMC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

- AND ..SUBTRACT B, F#1 I, F=22

14 DEC 79 PAGE 86

REF

PAGE \*\*\*\*\*

1573	*	1	..BIU(CA))(15)-Y -> BIU(CA))(15)
1574	*	2	CALL OP_READ(32_REG(1), "DISPLACE=0")
1575	*	3	CALL GET_BREG(U(A), REPEAT_ACCUMULATOR)
1576	*	4	32_REG(3) := REPEAT_ACCUMULATOR(15)
1577	*	5	32_REG(3) := 32_REG(3) - 32_REG(1) ..DIVES COMPLEMENT SUBTRACTION.
1578	*	6	REPEAT_ACCUMULATOR(15) := 32_REG(3)
1579	*	7	CALL PUT_BREG(U(A), REPEAT_ACCUMULATOR)
1580	*	8	RETURN

NSUC AN/JUVK-7 (CP)  
CP INSTRUCTION SET

\_SB - STORE B FMT I, F=23

RUR  
PAGE

1582	*	*	1	.. BLUCA)I(15) -> Y
1583	34	*	2	CALL GET-BREG(UCA), REPEAT_ACCUMULATOR
1584	*	*	3	REPEAT_ACCUMULATOR == REPEAT_ACCUMULATOR(15)
1585	15	*	4	CALL OP_STORE(PEPEAT_ACCUMULATOR, "DISPLACE" 0) ..REPEAT_ACCUMULATOR -> MEMORY.
1587	*	*	5	RETURN

14 DEC 79 PAGE 66

**MSMC** ANUVAK-7 (CP) CP INSTRUCTION SET

422 F. S. STONE AND J. L. TAYLOR

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 89

\_SIB .00 STORE A & INDEX B FMT I, F=25

REF PAGE \*\*\*\*\*  
1594 \* 1 /\* ALU(A) \*/ -> Y, B(U(C)) /\* BLU(B) \*/+1  
1595 \* 2 SET SPR\_PRIVILEGED\_INSTRUCTION\_INDICATOR  
1596 \* 3 CALL\_SA  
1597 \* 4 CALL\_GET\_BREGIU(CB), 32\_REG(11)  
1598 \* 5 32\_REG(115) /\* 32\_REG(11) {15} \*/+1  
1599 \* 6 CALL\_PUT\_BREGIU(CB), 32\_REG(11)  
1600 \* 7 RETURN

NSWC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

-SNA •• STORE NEGATIVE FHI I, F=26

REF PAGE \*\*\*\*\*

1602	*	1	** (NOT_AIU(CA)) -> Y
1603	*	2	CALL GET_AREG(U(C)), REPEAT_ACCUMULATOR
1604	*	3	REPEAT_ACCUMULATOR ** (NOT_Y REPEAT_ACCUMULATOR
1605	*	4	CALL OP_STORE(REPEAT_ACCUMULATOR, _DISPLACE=0)
1606	*	5	RETURN

14 DEC 79 PAGE 90

14 DEC 79 PAGE 91  
AN/UYK-7 (CP)  
CP INSTRUCTION SET

-SM --STORE MAGNITUDE FMT I, F=27

REF

PAGE \*\*\*\*\*

```
1608    33 * 1 CALL GET_AREGIUTA),REPEAT_ACCUMULATOR,  
1609    * 2 IF REPEAT_ACCUMULATOR(31,1) = 1, THEN  
1610    * 3 REPEAT_ACCUMULATOR != .NOT. REPEAT_ACCUMULATOR  
1611    * 4 ENDIF  
1612    15 * 5 CALL OP_STORE#REPEAT_ACCUMULATOR, "DISPLAY=0"  
1613    * 6 RETURN  
* *****
```

NSMC

A<sub>N</sub>/UYK-7 (CP)  
CP INSTRUCTION SET

-B2 (\_BS) ..CLEAR & SET BIT, FMT I, F=32 E F=33

REF

PAGE \*\*\*\*\*

```
1615 * 1 32_REG(2) /* UCAK) ..SAVE K FIELD & FORCE FULL WORD.  
1616 * 2 U(K) := FULL WORD TYPE  
1617 14 * 3 CALL OP_REPEAT_ACCUMULATOR, "DISPLACE=0"  
1618 * 4 IF U(F) EVEN, THEN  
1619 * 5 REPEAT_ACCUMULATOR(32_REG(2){4},13) := 0  
1620 * 6 ELSE ..SET 8 BIT INSTRUCTION.  
1621 * 7 REPEAT_ACCUMULATOR(32_REG(2){4},13) := 1  
1622 * 8 ENDIE  
1623 * 9 CALL_REPLACE(REPEAT_ACCUMULATOR) ..UPDATE IN MEMORY.  
1624 * 10 U(K) := 32_REG(2){2}  
1625 * 11 RETURN  
*****
```

NSWC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 93

-RA [C \_RAH] ..REPLACE ADD (SUBTRACT), FMT I, F=34 & F=36

REF PAGE \*\*\*\*\*  
1627 \* 1 CALL DP\_READ (REPEAT\_ACCUMULATOR, "DISPLACEMENT") ..FETCH OPERAND (Y) UNDER K DESIGNATOR CONTROL.  
1629 \* 2 CALL GET\_AREC (U(A), 32\_REG(1))  
1630 \* 3 IF U(F)=0,34, "REPLACE ADD", THEN  
1631 \* 4 REPEAT\_ACCUMULATOR := REPEAT\_ACCUMULATOR + 32\_REG(1) ..ONES COMPLEMENT ADDITION.  
1632 \* 5 ELSE ..REPLACE SUBTRACT (F=36).  
1633 \* 6 REPEAT\_ACCUMULATOR := REPEAT\_ACCUMULATOR - 32\_REG(1) ..ONES COMPLEMENT SUBTRACTION.  
1634 \* 7 ENDIF  
1635 \* 8 UPDATE FIXED POINT OVERFLOW (ASR(3,1))  
1636 \* 9 CALL UPDATE\_REPLACE (REPEAT\_ACCUMULATOR, U(A)+1)  
1637 \* 10 RETURN  
\*\*\*\*\*

-RI (E \_RD) ..REPLACE\_INCREMENT (DECREMENT), FMT I, F=35 E F=37

REF

PAGE \*\*\*\*\*  
\*\*\*\*\*  
1641 14 \* 1 CALL OP\_READ (REPEAT\_ACCUMULATOR, "DISPLACE=0")  
1642 \* 2 IF U(F)>Q+35\* "REPLACE\_INCREMENT", THEN  
1643 \* 3 REPEAT\_ACCUMULATOR = REPEAT\_ACCUMULATOR + 1 ..ONES COMPLEMENT.  
1644 \* 4 ELSE .."REPLACE DECREMENT".  
1645 \* 5 REPEAT\_ACCUMULATOR = REPEAT\_ACCUMULATOR - 1 ..ONES COMPLEMENT.  
1646 \* 6 ENDIF  
1647 \* 7 UPDATE FIXED POINT OVERFLOW (ASR(3,1))  
1648 \* 8 CALL UPDATE\_REPLACE (REPEAT\_ACCUMULATOR, U(A))  
1649 \* 9 RETURN  
\*\*\*\*\*

AM/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 95

\*\*MULTIPLY A, PPT I, F=4C

BEE PAGE \*\*\*\*\*  
1651 \* 1 \*\* A(L(A))>Y => A(L(A)-1) E A(L(A))  
1652 \* 2 CALL GET\_AREG L(CA), 32\_PEG(2) \*\*GET MULTIPLICAND.  
1653 \* 3 CALL CP\_PEG(32\_REG(1), "DISPLACE=0) \*\*GET MULTIPLIER.  
1654 \* 4 SIGN\_INDICATOR = 32\_PEG(1)(31,1) XOR 32\_PEG(2)(31,1) ..REMEMBER SIGN FOR RESULT.  
1655 \* 5 CALL DC\_MULTIPLY ..DO ACTUAL MULTIPLICATION.  
1656 \* 6 JE SIGN\_INDICATOR =NEGATIVE", THEN \*\*NEGATE ANSWER.  
1657 \* 7 REPEAT\_ACUMULATOR =C000..NCT.REPEAT\_ACUMULATOR 32\_REG(1)=..NOT.32\_PEG(1)  
1658 \* 8 ENDIE  
1659 \* 9 CALL PUT\_AREG L(U(A)+1, REPEAT\_ACUMULATOR) ..STORE MOST SIGNIFICANT BITS.  
1660 \* 10 CALL PUT\_AREG L(U(A)), 32\_REG(1) ..STORE LEAST SIGNIFICANT BITS.  
1661 \* 11 RETURB  
1662 \* 12 \*\*\*\*\*

## DO\_MULTIPLY

```

REF PAGE
 6   1   ** MULTIPLY 32_REG(2) BY 32_REG(1) WITH RESULT IN REPEAT_ACCUMULATOR C 32_REG(1),
1669  * 2   * 32_REG(1) * 32_REG(2). THE UT-7 ALGORITHM WAS PER TUKR-7 LEARNER'S GUIDE IJIN 1973, PAGE 423).
1670  * 3   FCCE 32_REG(1) POSITIVE & 32_REG(2) NEGATIVE USING ONES COMPLEMENT OPERATIONS.
1671  * 4   REPEAT_ACCUMULATOR := 0; MS_CARRY := 0 *OFF*; COUNT := 16
1672  * 5   DC WHILE COUNT P0
1673  * 6   DC CASE 32_REG(1)(1,2)
1674  * 7   NO
1675  * 8   IF *B-CARRY, THEN
1676  * 9   REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-32_REG(2); MS_CARRY := 0
1677  * 10  *ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1678  * 11  EBDLE
1679  * 12  32_REG(1)=32_REG(1)+RL2; 32_REG(1)(31,2)=REPEAT_ACCUMULATOR(1,2)
1680  * 13  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1681  * 14  \1
1682  * 15  IF MS_CARRY, THEN
1683  * 16  REPEAT_ACCUMULATOR:=REPEAT_ACCUMULATOR-(32_REG(2)-RL1); MS_CARRY=0
1684  * 17  *ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1685  * 18  ELSE
1686  * 19  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-32_REG(2)) -ONES COMPLEMENT SUBTRACT.
1687  * 20  EBDLE
1688  * 21  32_REG(1)=32_REG(1)+RL2; 32_REG(1)(31,2)=REPEAT_ACCUMULATOR(1,2)
1689  * 22  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1690  * 23  \2\

1691  * 24  IF MS_CARRY, THEN
1692  * 25  REPEAT_ACCUMULATOR SIGN := REPEAT_ACCUMULATOR(31,1)
1693  * 26  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2); 32_REG(1)(31,2)=ONES COMPLEMENT ADD.
1694  * 27  32_REG(1)=32_REG(1)+RL1; RL2=32_REG(1)(31,2)-REPEAT_ACCUMULATOR(1,2)
1695  * 28  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1696  * 29  IF ACCUMULATOR SIGN = "NEGATIVE", THEN
1697  * 30  REPEAT_ACCUMULATOR(31,2) := RL1; ..SET TWO HIGH BITS ON.
1698  * 31  ENDIF
1699  * 32  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR-32_REG(2); RL1=1
1700  * 33  *ONES COMPLEMENT SUBTRACT, END AROUND BORROW FORCED.
1701  * 34  SUBTRACT END AROUND BORROW.
1702  * 35  32_REG(1)=32_REG(1)+RL2; 32_REG(1)(31,2)=REPEAT_ACCUMULATOR(1,2)
1703  * 36  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1704  * 37  RL1=32_REG(1)(31,2)-REPEAT_ACCUMULATOR(1,2)
1705  * 38  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1706  * 39  RL1=32_REG(1)(31,2)-REPEAT_ACCUMULATOR(1,2)
1707  * 40  REPEAT_ACCUMULATOR(31,2) := RL1; ..FORCE HIGH BITS TO 0 & 1.
1708  * 41  ENDIF
1709  * 42  EBDLE
1710  * 43  \3\
1711  * 44  IF MS_CARRY, THEN
1712  * 45  32_REG(1)=32_REG(1)+RL2; 32_REG(1)(31,2)=REPEAT_ACCUMULATOR(1,2)
1713  * 46  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1714  * 47  MS_CARRY = 1 *ON*
1715  * 48  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR+32_REG(2); 32_REG(1)(31,2)=TWO'S COMPLEMENT ADD.
1716  * 49  32_REG(1)=32_REG(1)+RL2; 32_REG(1)(31,2)=REPEAT_ACCUMULATOR(1,2)
1717  * 50  REPEAT_ACCUMULATOR := REPEAT_ACCUMULATOR(1,2)-REPEAT_ACCUMULATOR(1,2)
1718  * 51  EBDLE
1719  * 52  Ebdic **END CASE.
1720  * 53  COUNT := COUNT-1
1721

```

## DO\_DIVIDE (REF 64\_REG, 32\_REG)

```

REF          PAGE
1727        * 1   ** DIVIDE THE CONTENTS OF 64_REG BY THE CONTENTS OF 32_REG WITH TIME QUOTIENT LEFT IN
1728        * 2   ** 64_REG(31,32) AND THE REMAINDER LEFT IN 64_REG(63,32). THIS USES THE UYK-7 ALGORITHM
1729        * 3   ** AS FOUND IN THE UYK-7 LEARNER'S GUIDE (JAN 1973) PAGES 435-442.
1730        * 4   ** (1) 64_REG = 64-BIT REFERENCE REGISTER.
1731        * 5   ** (2) 32_REG = 32-BIT DIVISOR.
1732        * 6   JE 32_REG = -0, THEN
1733        * 7   32_REG = 0 ..NEGATIVE ZERO IS A SPECIAL CASE.
1734        * 8   ENDIF
1735        * 9   SIGN_IND := 64_REG(63,1).XOR.32_REG(1,1) ..SIGN CORRECTION FLAG.
1736        * 10  SIGN_DIVIDEND := 64_REG(63,1) ..SAVE SIGN OF ORIGINAL DIVIDEND.
1737        * 11  FORCE 64_REG "DIVIDEND" AND 32_REG "DIVISOR" POSITIVE (ONES COMPLEMENT)
1738        * 12  COUNT := 32
1739        * 13  DO WHILE COUNT > 0
1740        * 14  64_REG := 64_REG.LL.1
1741        * 15  CALL DEVICE_COMPARE16$REG,32_REG)
1742        * 16  COUNT := COUNT-1
1743        * 17  ENDDO ..END WHILE.
1744        * 18  IF 64_REG(31,1) "QUOTIENT NEGATIVE", THEN ..OVERFLOW HAS OCCURRED.
1745        * 19  ASR(31,1) "OVERFLOW" := 1
1746        * 20  ELSE
1747        * 21  ASR(3,1) "OVERFLOW" := 0
1748        * 22  ENDIF
1749        * 23  IF SIGN_IND, THEN
1750        * 24  64_REG(31,32) := -NOT.64_REG(31,32) ..CORRECT SIGN OF QUOTIENT.
1751        * 25  ENDIF
1752        * 26  IF SIGN_DIVIDEND, THEN
1753        * 27  64_REG(63,32) := -NOT.64_REG(63,32) ..MAKE REMAINDER THE SAME SIGN AS THE ORIGINAL DIVIDEND.
1754        * 28  ENDIF
1755        * 29  RETURN

```

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 96

-0 .. DIVIDE A FRT I, F=41

REF PAGE \*\*\*\*\*

```
1761   *    1    .. (A(U(A+1)),A(U(A))) / V -> A(U(A)) ; REMAINDER -> A(U(A+1))
1762   14   *    2    CALL OP_PEA0(32,_REG1),_DISPLAY=0) .. GET DIVISOR.
1763   33   *    3    CALL GET_AREG(U(A),64,_REG(2)(3),32) .. GET 64-BIT DIVIDEND.
1764   33   *    4    CALL GET_AREG(U(A+1),64,_REG(1)(63,32))
1765   97   *    5    CALL DO_DIVIDE (64,_REG(1), 32,_REG(1)) .. DO THE ACTUAL DIVISION.
1766   36   *    6    CALL PUT_AREG(U(A+1),64,_REG(1)(63,32)) .. STORE REMAINDER.
1767   36   *    7    CALL PUT_AREG(U(A),64,_REG(2)(3),32) .. STORE QUOTIENT.
1768   *    8    RETURN
```

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 96

-BC .. COMPARE BIT TG ZERO, FMT I F=42

REF PAGE

```
1770 * 1 J = U(CAK)(4) . IGNORE BIT 25 => BIT * IS MODULO 32.  
1771 * 2 U(k) = 3 .. FORCE FULL WORD TYPE OPERAND FETCH.  
1772 * 3 CALL OPREAD(32_REG(1), "DISPLACEMENT"01  
1773 * 4 TO 32_REG(1)(J,1) = 0 "BIT CLEAR", THEN  
1774 * 5 ASR(2,1) = 1 .. INDICATE EQUAL.  
1775 * 6 ELSE .. BIT SET.  
1776 * 7 ASR(2,1) = 0 .. INDICATE NOT EQUAL,  
1777 * 8 ENDIF  
1778 * 9 U(CAK)(4) = J .. RESTORE THE ORIGINAL AK FIELD.  
1779 * 10 RETURN  
*
```

NSUC AN/ULR-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 100

-CMI ==COMPARE INDEX INCREMENT. FMT 1 F=43

REF

PAGE

```
1781    34 *   1      CALL GET_BREG(A),32_REG(11)
1782    14 *   2      CALL CP_PREAD(32_REG(2),"DISPLACEMENT")
1783    *   3      IF 32_REG(11)(15,16) >= 32_REG(2), THEN
1784    *   4          ASR(CJ) := 1 ..INDICATE OUTSIDE LIMITS.
1785    *   5          32_REG(11)(15,16) := 0 ..CLEAR THE INDEX REGISTER.
1786    *   6      ELSE
1787    *   7          ASR(0) := 0 ..INDICATE WITHIN LIMITS.
1788    *   8          32_REG(11)(15,16) := 32_REG(11)(15,16) + 1 ..INCREMENT THE INDEX REGISTER.
1789    *   9      ENDIF
1790    37 *   10     CALL PUT_BREG(A),32_REG(11)
1791    *   11     RETURN
```

NSUC AN/UTK-7 (CP)  
CP INSTRUCTION SET

-C . . .COMPARE, FMT 1, F=44

14 DEC 79 PAGE 101

REF PAGE

1793	1	" COMPARE AL(0) ; OPERAND, SET COMPARE DESIGNATORS IN ASR.
1794	2	CL : GET_AREG (LCA), 32_REG(1)
1795	3	CALL OPREAD (32_REG(2), "DISPLACE=0")
1796	4	CALL SET_CD1 (32_REG(1), 32_REG(2))
1797	5	RETURNS

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 102

\_CL ••COMPARE LIMITS, FMT I, F=45

REF PAGE \*\*\*\*\*  
1799 \* 1 \* A(U(A))> A(U(A)+1) : OPERAND. SET COMPARE DESIGNATORS IN ASR.

1800 \* 2 \* CALL GET\_AREG (U(A)), 32\_REG(2)

1801 \* 3 \* CALL GET\_AREG (U(A)+1), 32\_REG(3)

1902 \* 4 \* CALL OP\_READ (32\_REG(1), "DISPLAY E=0")

1803 \* 5 \* CALL SET\_CD2 (32\_REG(1), 32\_REG( ), 32\_REG(3))

1804 \* 6 \* RETURN

NSWC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 103

\_CM ..COMPARE MASKED, FMT I, F=46

REF PAGE \*\*\*\*\*

```
1606   * 1    /* A(U(A))+1; 1 (A(U(A)).AND.OPERAND). SET COMPARE DESIGNATORS IN ASR.
1807   * 2    CALL GET_AREG (U(A)+1, 32_REG(1))
1808   * 3    CALL GET_AREG (U(A), 32_REG(2))
1809   * 4    CALL OP_READ (32_REG(3), "DISPLACE=0")
1810   * 5    32_REG(2) = 32_REG(2) -A. 32_REG(3)
1811   * 6    CALL SET_CD1 (32_REG(1), 32_REG(2))
1812   * 7    RETURN
*****
```

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 104

CG ..COMPARE GATED, FMT 1, F=47

REF PAGE \*\*\*\*\*

1816	*	1	** ABS(OPERAND - A(U[A])+1) - A(U[A]+1). SET COMPARE DESIGNATORS IN ASR.
1815	*	2	CALL OP READ (32_REG(1), -DISPLACE=0)
1816	*	3	CALL GET_AREG (U[A], 32_REG(3))
1817	*	4	32_REG(1) := ABSOLUTE VALUE OF (32_REG(1) - 32_REG(3))
1818	*	5	CALL GET_AREG (U[A]+1, 32_REG(2))
1819	*	6	CALL SET_CDI (32_REG(1), 32_REG(2))
1820	*	7	RETBN

JEP (JOP) ..JUMP EVEN (ODD) PARITY, F#11, F=50 0 & F=50 1

REF PAGE \*\*\*\*\*  
1822 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION\_INDICATOR ..INDICATE PRIVILEGED IF SPR{16,1} SET AND UC11 SET.  
1824 \* 2 CALL JUMP\_ADDRESS ("DISPLACE"o, OPERAND\_S, OPERAND\_DISPLACEMENT)  
1825 \* 3 ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS. OPERAND BASE  
1826 \* 4 ..REGISTER AND OPERAND DISPLACEMENT ARE RETURNED.  
1827 \* 5 CALL GET\_AREG [UC4], 32\_REG(11)  
1828 \* 6 CALL GE\_AREG [UL4]+1, 32\_REG(2)  
1829 \* 7 32\_REG(11) := 32\_REG(11) .A. 32\_REG(2)  
1830 \* 8 1 := 0  
1831 \* 9 J := 0  
1832 \* 10 DO WHILE I < 32  
1833 \* 11 J := J + 32\_REG(11){I,1}  
1834 \* 12 I := I + 1  
1835 \* 13 ENDDO  
1836 \* 14 J := J .X. UF2  
1837 \* 15 IF J(0) = 0, THEN ..TEST FOR JOP/ODD OR JEP/EVEN  
1838 \* 16 CALL DD\_JUMP (OPERAND\_S, OPERAND\_DISPLACEMENT)  
23 \* 16 ..ENDIF  
1839 \* 17 RETURN  
1840 \* 18 \*\*\*\*\*

NSWC  
ANSIYR-7 (CP)  
CP INSTRUCTION SET

14 DEC 76 PAGE 206

-DJZ ...JUMP DOUBLE PRECISION ZERO, FMT III F=50 2

REF	PAGE	1	SET SPR_PRIVILEGED_INSTRUCTION .. INDICATE PRIVILEGED IF SPR(16)1) SET AND U(11) SET.
1842	*	2	CALL JUMP ADDRESS("DISPLACE")0, OPERAND_SS, OPERAND_DISPLACEMENT)
1845	13	3	..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1846	*	4	CALL GET_OPERAND(64, REG(1){31,32})
1847	33	5	CALL GET_OPERAND(1+1,64, REG(1){63,32})
1848	*	6	IF 64_REG(1) = 0, THEN ..JUMP ON POSITIVE ZEPIC ONLY!
1849	23	7	CALL CO_JUMP (OPERAND_SS, OPERAND_DISPLACEMENT)
1850	*	8	ENDIF
1851	*	9	RETURN

407-7 (CPI)  
DC\_INSTRUCTION SET

16 DEC 74 PAGE 107

-JBLZ . . . JBLZ DOUBLE PRECISION NOT ZERO, FMT III F=50 3

REF PAGE

```
l851   1   1 SET SP8_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR{16,1} SET AND UC{1} SET.  
l852   13   2 CALL JUMP_ADDRESS=C-DISPLACE=0, OPERAND_S, OPERAND_DISPLACEMENT;  
l853   13   3 ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.  
l854   33   4 CALL GET_AREG1(LA),64-REG1){31,32};  
l855   33   5 CALL GET_AREG1(LA)+1,64-REG1){63,32};  
l856   33   6 IF 66_REG11 <> 0, THEN ..TEST POSITIVE ZERO ONLY!  
l857   23   7 CALL DC_JUFP (OPERAND_S, OPERAND_DISPLACEMENT);  
l858   23   8 ENDIF  
l859   23   9 RETURN
```

MSWC AN11414-7 (CF)  
CP INSTRUCTION SET

14 DEC 70 PAGE 106

-FS11\_JP, -JN, -JT, -JNL FMT III F-51 0 THRU 51 3

REF  
PAGE \*\*\*\*\*  
  
1864 \* 1 SET\_SFR\_PRIVILEGED\_INSTRUCTION /\*INDICATE PRIVILEGED IF SPP(16,1) SET AND U(1) SET.  
1865 \* 2 CALL\_JUMP\_ADDRESS(CS,DISPLACE\_0, OPERAND\_S, COPERAND\_DISPLACEMENT).  
1866 \* 3 --OR OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.  
1867 \* 4 CALL\_GET\_AREG(U(1),32{REG11})  
1868 \* 5 JUMP\_IS\_FALSE  
1869 \* 6 20 -CASE\_U(CF3) CF  
1870 \* 7 \0\ IF 32{PEG11}{31,1} = 0 "-POSITIVE", THEN JUMP\_IS\_TRUE ENDIF  
1871 \* 8 \1\ IF 32{PEG11}{31,1} = 1 "-NEGATIVE", THEN JUMP\_IS\_TRUE ENDIF  
1872 \* 9 \2\ IF 32{PEG11} = 0 "-POSITIVE ZERO", THEN JUMP\_IS\_TRUE ENDIF  
1873 \* 10 \3\ IF 32{PEG11} <> 0 "-NOT POSITIVE ZERO", THEN JUMP\_IS\_TRUE ENDIF  
1874 \* 11 ENDDO  
1875 \* 12 JE JUMP\_IS\_TRUE  
1876 \* 13 CALL\_OU\_JUMP (COPERAND\_S, OPERAND\_DISPLACEMENT)  
1877 \* 14 EndIF  
1878 \* 15 RETURN  
1879 \* 16 \*\*\*\*\*

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

\_LBJ ..LOAD B AND JUMP, FMT III, F=52 C

14 DEC 79 PAGE 109

REF PAGE \*\*\*\*\*  
1881 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION ..INDICATE PRIVILEGED IF SPR{16,1} SET AND UC10 SET.  
1882 \* 2 IF UC10 = 0, THEN  
1883 \* 3 RETURN ..NC OPERATION.  
1884 \* 4 ELSE  
1885 \* 5 32\_REG11{19,3} := P{S3} ..P REGISTER BASE REGISTER DESIGNATOR.  
1886 \* 6 32\_REG11{15,16} := P{D3} ..P REGISTER DISPLACEMENT (ALREADY INCREMENTED).  
1887 \* 7 CALL PUT\_BREG11A,32(REG11)  
1888 \* 8 CALL PUT\_BREG11B,32(REG11)  
1889 \* 9 CALL JUMP\_ADDRESS("DISPLACE0, OPERAND\_S, OPERAND\_DISPLACEMENT")  
1890 \* 10 CALL JUMP\_TYPE\_INSTRUCTIONS.  
1891 \* 11 .DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.  
1892 \* 12 CALL DO\_JUMP (COPERAND\_S, OPERAND\_DISPLACEMENT)  
1893 \* 13 ENDIF  
1894 \* 14 RETURN  
\*\*\*\*\*

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

-JENZ ..INDEX JUMP B, FMT III, F=52 1

REF  
PAGE

```
1096    *   1   SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.  
1096    *   2   CALL GET_BREG(U(A),32(REG(1))  
1099    *   3   IF 32(REG(1){15,16} <> 0, THEN  
1099    *   4       32_REG(1){15,16} = 32_REG(2){15,16} - 1 ..DECREMENT THE INDEX REGISTER.  
1902    *   5   CALL PUT_BREG(UA),32(REG(1))  
1903    *   6   CALL "Jump_Address"(“DISPLACE=0, OPERAND_S, OPERAND_DISPLACEMENT)  
1904    *   7   ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.  
1905    *   8   CALL DO_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)  
1906    *   9   EDDIE  
1907    *  10   RETURN
```

14 DEC 79 PAGE 110

MSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

JIS • JUMP SY & B, FMT III F=52 2

14 DEC 79 PAGE 111

REF PAGE  
\*\*\*\*\*  
1909 34 \* 1 CALL GET\_BREG(US), 32\_REG(11)  
1910 \* 2 PCS) \* 32\_REG(11)(19,3) \*P(S) GETS NEW BASE REGISTER DESIGNATOR.  
1911 \* 3 PCD) \* 32\_REG(11)(15) + U(SY) ..PCD) GETS NEW DISPLACEMENT.  
1912 \* 4 32\_REG(11) \* P(D)  
1913 32 \* 5 CALL ADD\_SAP(S), 32\_REG(11) \*JUMPED TO ADDRESS GOES TO 32\_REG(11).  
1914 17 \* 6 CALL BPR\_CHECK(32\_REG(11), OPERAND) ..DO OPERAND CHECK ON JUMPED TO ADDRESS.  
1916 \* 7 RETURN  
\*\*\*\*\*

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

-JL ..JUMP LOWER, FAT III, F=52 3

REF

PAGE

```

1918   1 SET SPR_PRIVILEGED_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16..11) SET AND UC(1) SET.
1920   2 -- NOTE: BPR CHECKS & ILLEGAL INSTRUCTION CHECKS ARE DONE IN THIS ROUTINE TO
1922   3 --SUPPLEMENT NORMAL I SEQUENCE HANDLING OF THE LOWER HALF WORD INSTRUCTION.
1924  13  4 CALL JUMP_ADDRESS(DISPLACE), OPERAND_S, OPERAND_DISPLACEMENT)
1925   5 --DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
1926   6 PCS) := OPERAND_S --SET UP BASE (SI DESIGNATOR FOR JUMPED TO INSTRUCTION.
1928   7 P(DI) := OPERAND_DISPLACEMENT --SET UP DISPLACEMENT FOR JUMPED TO INSTRUCTION.
1930   8 32(REG1) := P(CJ) --COPY THE DISPLACEMENT FOR CHECKS.
1931   9 ASR(CL1) = 1 --SET LOWER HALF WORD INDICATOR.
1932  10 CALL ADD_SIUC(S1,32_REG1) --COMPUTE ABSOLUTE ADDRESS OF JUMPED TO INSTRUCTION.
1934  11 CALL BPR_CHECK(32_REG1),INSTRUCTION)
1935  12 --DO BPR CHECKS, PECDLAR TO LOWER HALF WORD INSTRUCTIONS ARRIVED AT BY JL INSTRUCTION.
1937  13 CALL MEMORY_READ(32_PEG(11),32_REG12),UF53,INSTRUCTION) --FETCH INSTRUCTION.
1939  14 -- NOTE: THE INITIAL CALL TO OP_READ INSURES THE INSTRUCTION IS AT A
1940  15 --VALID MEMORY LOCATION, THUS NO INTERRUPT SHOULD OCCUR.
1941  16 IE 32_REG(2)(15,2) => B(J1), THEN --CHECK FOR VALID HALF-WORD INSTRUCTION CANDIDATE.
1943  17 CALL GENERATE_SYNCHRONOUS_INTERRUPT(p=-1, OP ILLEGAL INSTRUCTION)
1944  18 END
1945  19 RETURN

```

NSUC AN/UKK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 113

-J530 --JUMP ON ASR OVERFLOW DESIGNATOR (JNF, JOF), FMT III F=53 0

REF

PAGE \*\*\*\*\*  
\*\*\*\*\*  
1947 \* 1 SET SPR\_PRIVILEGED\_INSTRUCTION --INDICATE PRIVILEGED IF SPR(16,1) SET AND U(I) SET.  
1949 13 \* 2 CALL JUMP\_ADDRESS("G", OPERAND\_S, OPERAND\_DISPLACEMENT)  
1950 \* 3 ..DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS..  
1951 \* 4 J I\* ASR(3,1) "OUTSIDE/WITHIN" .X. U(A)  
1952 \* 5 ASR(3,1) == 0 --CLEAR ASR OVERFLOW INDICATOR.  
1953 \* 6 IF J(0,1)=0, THEN  
1954 23 \* 7 CALL DC\_JUMP(OPERAND\_S, OPERAND\_DISPLACEMENT)  
1955 \* 8 ENDIF  
1956 \* 9 RETURN  
\*\*\*\*\*

-J531 ...JUMP ON ASR COMPARE DESIGNATOR, FMT LIX F=53 1

```

REF PAGE *****
1958   * 1   ** THIS ROUTINE EXECUTES THE FOLLOWING INSTRUCTIONS : JNE, JE, JGE, JLT, JLE, JNE, JW.
1960   * 2   SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPR{16,1} SET AND U{1} SET.
1962   * 3   CALL JUMP_ADDRESS="O", OPERAND_S, OPERAND_DISPLACEMENT
1963   * 4   DO OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS
1964   * 5   JUMP := FALSE ..INITIALIZE THE JUMP FLAG.
1965   * 6   EC CASE U{A} OF
1966   * 7   * VCV V11 ...JNEP JE.
1967   * 8   J := U{AD} .X. ASR{2,1} "EQUAL/UNEQUAL"
1968   * 9   IF J{0,1}=0, THEN
1969   * 10  JUMP := TPUF
1970   * 11  ENDIE
1971   * 12  ** V21 ** JG.
1972   * 13  IE ASR{1,1},J1 "GREATER THAN OR EQUAL" .A. ASR{2,1}=0 "UNEQUAL", THEN
1973   * 14  _JUMP := TRUE
1974   * 15  ENDIE
1975   * 16  ** V31 V41 ** JGE, JLT.
1976   * 17  J := U{A} .X. ASR{1,1} "GREATER THAN OR EQUAL/LESS THAN"
1977   * 18  IF J{G,1}=0, THEN
1978   * 19  _JUMP := TRUE
1979   * 20  ENDIE
1980   * 21  ** V51 ** JLE.
1981   * 22  IE ASR{1,1}=0 "LESS THAN" .W. ASR{2,1}=1 "EQUAL", THEN
1982   * 23  _JUMP := TRUE
1983   * 24  ENDIE
1984   * 25  ** V61 V71 ** JNE, Jb.
1985   * 26  J := U{A} .X. ASR{0,1} "OUTSIDE/WITHIN"
1986   * 27  IF J{G,1}=1, THEN
1987   * 28  _JUMP := TRUE
1988   * 29  ENDIE
1989   * 30  ENDOD
1990   * 31  IE _JUMP = TRUE, THEN
1991   * 32  CALL DC_JUMP(OPERAND_S, OPERAND_DISPLACEMENT),
1992   * 33  ENDIE
1993   * 34  RETURN

```

J532 ..RETURN JUMPS, FAT III F=53 2

REF PAGE

```

1995 * 4   SET SPR_PRIVILEGED_INSTRUCTION ..INDICATE PRIVILEGED IF SPP{16,1} SET AND U(Y) SET.
1997 * 2   .MATCH := FALSE ..INITIALIZE THE MATCH FLAG.
1998 * 3   JUMP_STOP := UCA(2,1) ..HIGH ORDER BIT OF "A" FIELD.
1999 * 4   IE_JUMP_STOP := AND( ASRC19,6) > 0 "TASK MODE", THEN
2000 * 5   GENERATE SYNCHRONOUS INTERRUPT(P=1), PRIVILEGED INSTRUCTION VIOLATION!
2001 * 6   ..ABORTS THE INSTRUCTION.
2002 * 7
2003 * 8   IF UCA(1,2), THEN ..CHECK APPROPRIATE JUMP SWITCH.
2004 * 9   IE_JUMP_STOP, THEN
2005 * 10  IF STOP_SWITCH(UCA) ==1, THEN
2006 * 11  ..MATCH := TRUE
2007 * 12
2008 * 13
2009 * 14  IF JUMP_SWITCH(UCA) ==1, THEN
2010 * 15  ..MATCH := TRUE
2011 * 16
2012 * 17
2013 * 18
2014 * 19  ELSE ..O-> ALWAYS JUMP
2015 * 20  ..MATCH := TRUE
2016 * 21  32_REG11(131,15) := P(S)
2017 * 22  32_REG11(15) := P(0)
2018 * 23  CALL OP_STORE(32_REG(1), "DISPLACE=0") ..STORE P.
2019 * 24  CALL JUMP_ADDRESS("DISPLACE=1", OPERAND_S, OPERAND_D, DISPLACEMENT)
2020 * 25  ..00 OPERAND CHECKS FOR JUMP TYPE INSTRUCTIONS.
2021
2022 * 26  IF JUMP_STOP ..OR.. L_MATCH = TRUE, THEN
2023 * 27  CALL DO_JUMP(OPERAND_S, OPERAND_D, DISPLACEMENT)
2024 * 28  IF JUMP_STOP ..AND.. L_MATCH = TRUE, THEN
2025 * 29  ..SUSPEND PROCESSING (INDICATING STOP ACTIVE) UNTIL RESTARTED
2026 * 30
2027 * 31
2028 * 32
2029

```

J533 .0 MANUAL JUMPS, FAT III, F=53 3

REF  
PAGE

```
2030   * 2 SET SPR_PRIVILEGED_INSTRUCTION_INDICATOR
2031   * 2 CALL_JUMP_ADDRESS {"DISPLACE=0, OPERAND_S, OPERAND_DISPLACEMENT"}
2032   * 3 _MATCH = FALSE /*ASSUME NO JUMP.
2033   * 4 JUMP_STOP = UCA){2,1} /*I.E. HIGH BIT OF A FIELD.
2034   * 5 IF_JUMP_STOP AND ASR(19,4)=0 "TASK MODE", THEN
2035   * 6 CALL GENERATE_SYNCHRONOUS_INTERRUPT {"P=1", PRIVILEGED_INSTRUCTION_VIOLATION}
2036   * 7 /*ABORT THIS INSTRUCTION.
2037   * 8 ENDIE
2038   * 9 IF_UCA){1,2}, THEN /*CHECK APPROPRIATE JUMP SWITCH.
2039   * 10 IF_JUMP_STOP, THEN
2040   * 11   IF_STOP_SWITCH(UCA){1,1}, THEN
2041   * 12     _MATCH = TRUE
2042   * 13   ENDIE
2043   * 14 ELSE
2044   * 15   IF_JUMP_SWITCH(UCA){1,1}, THEN
2045   * 16     _MATCH = TRUE
2046   * 17   ENDIE
2047   * 18 ENDIE
2048   * 19 ELSE /*0 */ ALWAYS JUMP.
2049   * 20   _MATCH = TRUE
2050   * 21 ENDIE
2051   * 22 IF (JUMP_STOP .OR. (_MATCH=TRUE)), THEN
2052   * 23   CALL_Q0_JUMP (OPERAND_S, OPERAND_DISPLACEMENT)
2053   * 24   _E (JUMP_STOP .AND. (_MATCH=TRUE)), THEN
2054   * 25     SUSPEND PROCESSING (INDICATING STOP ACTIVE) UNTIL RESTARTED
2055   * 26   ENDIE
2056   * 27 ENDIE
2057   * 28 RETURN
```

LCT (6 LCI) .-LOAD CMR, FHI I, F=54 &amp; F=55

REF  
PAGE

```
*****  
* 1   * TREATED AS A FORMAT III INSTRUCTION IN THAT U(K) DOESN'T AFFECT OPERAND.  
2060  * 2   * 32_REG(2) := U(K) * ISOLATE AND SAVE "AK" FIELD.  
2062  * 3   * IF U(F) GOOD, THEN ..INTERRUPT CONTROL MEMORY REFERENCED.  
2063  * 4   * 32_REG(2) := 32_REG(2)0*100* ..DISPLACE TO INTERRUPT ADDRESSES.  
2064  * 5   * ENDIE  
2065  * 6   * IF (32_REG(2)>0*17) *OR. REPEAT IN PROGRESS) *AND. ASRL19,42*0 "TASK MODE", THEN  
2066  * 7   * CALL GENERATE_SYNCHRONOUS_INTERRUPT {"P-"}, PRIVILEGED INSTRUCTION VIOLATION)  
2068  * 8   * 11   * ABORT THIS INSTRUCTION.  
2070  * 9   * ENDIE  
2071  * 10  * U(K) := 3 "FULL WORD"  
2072  * 11  * CALL OP_READ (REPEAT_ACCUMULATOR, "DISPLACE=0")  
2073  * 12  * CMR(32_REG(2)) := REPEAT_ACCUMULATOR ..UPDATE CONTROL MEMORY CONTENTS.  
2074  * 13  * IF REPEAT_IN_PROGRESS, THEN  
2075  * 14  * 32_REG(2) := 32_REG(2)+1 ..INCREMENT AK FIELD.  
2076  * 15  * ENDIE  
2077  * 16  * U(AK) := 32_REG(2){5}  
2078  * 17  * RETURN  
2079  * 18  *
```

\*\*\*\*\*

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 110

SCT (E SCI) ..STORE CMB, FMT 1, F=56 C F=57

REF PAGE \*\*\*\*\*  
2081 \* 1 .REPEATED AS A FORMAT III INSTRUCTION IN THAT UC(K) DOESN'T AFFECT OPERAND.  
2083 \* 2 32\_REG(2) := UC(AK) .ISOLATE AND SAVE "AK" FIELD.  
2084 \* 3 IF UC(F) ODD, THEN .INCREMENT CONTROL MEMORY REFERENCED.  
2085 \* 4 32\_REG(2) := 32\_REG(2)+0'100' .DISPLACE TO INTERRUPT ADDRESSES.  
2086 \* 5 ENDIF  
2087 \* 6 IE (32\_REG(2)>0'17' .OR. REPEAT\_IN\_PROGRESS) .AND. ASRL(9,4)=0 "TASK MODE", THEN  
2089 \* 7 CALL GENERATE\_SYNCHRONOUS\_INTERRUPT (=P=1, PRIVILEGED INSTRUCTION VIOLATION)  
2091 \* 8 .AEORT THIS INSTRUCTION.  
2092 \* 9 ENDIF  
2093 \* 10 REPEAT\_ACCUMULATOR := CRR(32\_REG(2)) ..CONTROL MEMORY CONTENTS GOES TO REPEAT\_ACCUMULATOR.  
2095 \* 11 UC(K) := 3 FULL WORD  
2096 \* 12 CALL OP\_STORE (REPEAT\_ACCUMULATOR, "DISPLACE"=0)  
2097 \* 13 IF REPEAT\_IN\_PROGRESS, THEN  
2098 \* 14 32\_REG(2) := 32\_REG(2)+1 .INCREMENT AK FIELD.  
2099 \* 15 ENDIF  
2100 \* 16 UC(AK) := 32\_REG(2)(5) ..RESTORE AK FIELD.  
2101 \* 17 RETURN  
\*\*\*\*\*

HSC\_6C \*\*STORE CMR IN A, FRT IV A, F=60

```

REF PAGE *****
*   1   ADR := U{AF5}  **EXTRACT CONTROL MEMORY ADDRESS (I.E. 6-BIT "A" & "F4" FIELDS).
2103 *   2   IF U(1) = "HSCI", THEN
2105 *   3       ADR := AUR + 01000  **DISPLACE TO INTERRUPT ADDRESSES.
2106 *   4   ENDIF
2107 *   5   IF ADR>=0120  **AND. ASRC10,4)=0 "TASK MODE", THEN
2108 *   6       CALL GENERATE_SYNCHRONOUS_INTERRUPT (*P=1, PRIVILEGED INSTRUCTION VIOLATION)
2109 *   7       **ABORT THIS INSTRUCTION.
2111 *   8   ENDIF
2112 *   9   CALL HALFWORD_TOGGLE **CALL AFTER INTERRUPT POSSIBILITY FOR ECP 97A.
2113 *   10  IF (ADR<>0137) **AND. ADR>=0130) OR. (ADR<>057) **AND. ADR>=0130), THEN
2114 *   11      32_REG11 := 0  **UNASSIGNED CONTROL MEMORY IS A SOURCE OF ZEROS.
2115 *   12  ELSE
2116 *   13      IF U{AF5}<>017  **AND. U{AF4}>011  **INDEX REGISTER REFERENCE", THEN
2117 *   14          32_REG11 := CMR(ADR){15} .EXTRACT ONLY 16-BITS.
2118 *   15  ELSE
2119 *   16      IF ADR<>0170  **AND. ADR>=0170  **ACTIVE STATUS REGISTER", THEN
2120 *   17          32_REG11 := CRR1701  **ALL 7X REFER TO ASR.
2121 *   18  ELSE
2122 *   19      IF ADR>=60  **BREAKPOINT REGISTER", THEN
2123 *   20          32_REG11 := CMR(60)  **ALL 6X REFER TO BREAKPOINT REGISTER.
2124 *   21      ELSE  **ALL OTHER REFERENCES.
2125 *   22          32_REG11 := CMR(ADR)
2126 *   23      ENDIF
2127 *   24  ENDIF
2128 *   25  ENDIF
2129 *   26  ENDIF
2130 *   27  CALL PUT_AREG (U(0), 32_REG11)
2131 *   28  RETURN
2132 *   29
2133 *   30

```

HLC\_61 ..LOAD CAR FROM A, FRT IV A, F=61

```

REF PAGE *****
2135 * 1 ADR := UCAF4) ..EXTRACT CONTROL MEMORY ADDRESS (I.E. b-811 =A= & =F4= FIELDS).
2137 * 2 IF UC1) "HLC1", THEN
2138 * 3 ADR := ADR + 0'100' ..DISPLACE TO INTERRUPT ADDRESSES.
2139 * 4 ENDIF
2140 * 5 IF ADR>=0'20' ..AND. ASR{19,4)=0 "TASK MODE", THEN
2141 * 6 CALL GENERATE SYNCHRONOUS_INTERRUPT (*P=1, PRIVILEGED INSTRUCTION VIOLATION)
2142 * 7 ..ABORT THIS INSTRUCTION.
2143 * 8 ENDIF
2144 * 9 CALL HALFWORD_TOGGLE ..CALL AFTER INTERRUPT POSSIBILITY FOR ECP 9TA.
2145 * 10 IF (ADR<=0'137' ..AND. ADR>=0'130') ..OR. (ADR<=0'57' ..AND. ADR>=0'30'), THEN
2146 * 11 RETURN ..NOT ACCESSIBLE, THUS NOOP.
2147 * 12 ENDIF
2148 * 13 CALL GET_AREG UC(0), 32_REG{11}
2149 * 14 IF UCAF4)<=0'17' ..AND. UCAF4)>=0'11' "INDEX REGISTER REFERENCE", THEN
2150 * 15 CRR(ADR){15) := 32_REG{11}{15) ..UPDATE ONLY LOW 16 BITS.
2151 * 16 ELSE
2152 * 17 CALL GET_AREG UC(0), 32_REG{11)
2153 * 18 IF UCAF4)<=0'17' ..AND. ADR>=0'70' "ACTIVE STATUS REGISTER", THEN
2154 * 19 ELSE
2155 * 20 IF ADR<=C 7 "BREAKPOINT REGISTER", THEN
2156 * 21 CRR{60) := 32_REG{1) ..ALL 6X REFER TO THE BREAKPOINT REGISTER.
2157 * 22 ELSE ..ALL OTHER REFERENCES.
2158 * 23 CRR(ADR) := 32_REG{1)
2159 * 24 ENDIF
2160 * 25 ENDIF
2161 * 26 ENDIF
2162 * 27 ENDIF
2163 * 28 RETURN
2164 * 29
2165 *

```

\_HLC {S \_MRZ &\_MRS} SHIFT LEFT CIRCULARLY (RIGHT LOGICAL)

REF

PAGE

```
*   1  *RIGHT ARITHMETIC), FMT IV B, F=62, F=64, F=66
2167  24  * 2  CALL GET_SHIFT_AMOUNT (UCA), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.
2168  33  * 3  CALL GET_AREG (UCA), 32_REG(1)
2170  20 CASE UCF) ..SHIFT ACCORDING TO OPERATION CODE.
2171  * 4
2172  * 5
2173  * 6
2174  * 7
2175  * 8
2176  * 9
2177  * 10
2178  * 11

*   1  *RIGHT ARITHMETIC), FMT IV B, F=62, F=64, F=66
2167  24  * 2  CALL GET_SHIFT_AMOUNT (UCA), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.
2168  33  * 3  CALL GET_AREG (UCA), 32_REG(1)
2170  20 CASE UCF) ..SHIFT ACCORDING TO OPERATION CODE.
2171  * 4
2172  * 5
2173  * 6
2174  * 7
2175  * 8
2176  * 9
2177  * 10
2178  * 11

*   1  *RIGHT ARITHMETIC), FMT IV B, F=62, F=64, F=66
2167  24  * 2  CALL GET_SHIFT_AMOUNT (UCA), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.
2168  33  * 3  CALL GET_AREG (UCA), 32_REG(1)
2170  20 CASE UCF) ..SHIFT ACCORDING TO OPERATION CODE.
2171  * 4
2172  * 5
2173  * 6
2174  * 7
2175  * 8
2176  * 9
2177  * 10
2178  * 11

*   1  *RIGHT ARITHMETIC), FMT IV B, F=62, F=64, F=66
2167  24  * 2  CALL GET_SHIFT_AMOUNT (UCA), SHIFT_COUNT) ..ACQUIRE 6-BIT SHIFT COUNT.
2168  33  * 3  CALL GET_AREG (UCA), 32_REG(1)
2170  20 CASE UCF) ..SHIFT ACCORDING TO OPERATION CODE.
2171  * 4
2172  * 5
2173  * 6
2174  * 7
2175  * 8
2176  * 9
2177  * 10
2178  * 11
```

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 122

-HDLIC (6 -HDRZ & -HCRS) ..SHIFT LEFT DOUBLE CIRCULAR 6

REF

PAGE \*\*\*\*\*

```
2180 * 1   **PIGMOLOGICAL & RIGHT ARITHMETIC, FMT IV B, F=63, F=65, F=67
2181 24 * 2   CALL GET_SHIFT_AMOUNT (U(M), SHIFT_COUNT)   **ACQUIRE 6-BIT SHIFT COUNT.
2182 33 * 3   CALL GET_AREG (U(A)+1, 64-REG1)(63-32)  **MOST SIGNIFICANT BITS.
2183 33 * 4   CALL GET_AREG (U(A), 64-REG1)             **LEAST SIGNIFICANT BITS.
2184 * 5   DE CASE U(F) 64-REG1(31)  **LEAST SIGNIFICANT BITS.
2185 * 6   DE CASE U(F) 64-REG1 ACCORDING TO OPERATION CODE
2186 * 7   1631 SHIFT 64-REG11 LEFT CIRCULAR BY SHIFT-COUNT
2187 * 8   1651 SHIFT 64-REG11 RIGHT LOGICAL BY SHIFT-COUNT
2188 * 9   1671 SHIFT 64-REG11 RIGHT ARITHMETIC BY SHIFT-COUNT
2189 * 10  END CASE.
2190 36 * 11  CALL PUT_AREG (U(A)+1, 64-REG1)(63-32)  **MOST SIGNIFICANT BITS.
2191 36 * 12  CALL PUT_AREG (U(A), 64-REG1)(31)  **LEAST SIGNIFICANT BITS.
2192 * 13  RETURN
* *****
```

NSWC AN/UYK-7 (CPI)  
CP INSTRUCTION SET

14 DEC 79 PAGE 123

-HSF ==SCALE FACTOR, FHT IN A, F=70 0

REF PAGE \*\*\*\*\*  
2194 \* 1 CALL GET\_AREG (UFA), 32\_REG(11)  
2195 \* 2 IF 32\_REG(11)(31,1), THEN  
2196 \* 3 SHIFT\_COUNT := (NUMBER OF LEFT JUSTIFIED ONE BITS IN 32\_REG(11)) - 1  
2197 \* 4 ELSE SHIFT\_COUNT := (NUMBER OF LEFT JUSTIFIED ZERO BITS IN 32\_REG(11)) - 1  
2198 \* 5 ENDIF  
2199 \* 6  
2200 \* 7 \*\*NOTE THAT ALL ONES OR ZEROS MEANS THAT SHIFT\_COUNT = 0\*\*  
2201 \* 8 IF SHIFT\_COUNT<037, THEN  
2202 \* 9 SHIFT LEFT CIRCULARLY 32\_REG(11) BY SHIFT\_COUNT  
2203 \* 10 CALL PUT\_AREG (UFA), 32\_REG(11) ..RESTORE SHIFTED VALUE.  
2204 \* 11 ENDIF  
2205 \* 12 IF ULA>>UCB, THEN  
2206 \* 13 CALL PUT\_AREG (UFB), SHIFT\_COUNT  
2207 \* 14 ENDIF  
2208 \* 15 CALL HALFWORD\_TOGGLE  
2209 \* 16 RETURN

\*\*\*\*\*

NSWC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 124

\_HDSF \*\* DOUBLE SCALE FACTOR, FMT IV A, F=70 1

REF

PAGE \*

```
2211    33 * 1 CALL GET_AREG(U(A)+1), 64_REG(1){63,32};  
2212    33 * 2 CALL GET_AREG(U(A), 56_REG(1){32});  
2213    * 3 IF 64_REG(1){C3,1}, THEN  
2214    * 4 SHIFT_COUNT := (# OF LEFT JUSTIFIED BYES) - 1  
2215    * 5 ELSE  
2216    * 6 SHIFT_COUNT := (# OF LEFT JUSTIFIED ZEPUS) - 1  
2217    * 7 ENDIF  
2218    * 8 IF SHIFT_COUNT < 0'77', THEN  
2219    * 9 SHIFT LEFT CIRCULARLY {64_REG(1), SHIFT_COUNT}  
2220    36 * 10 CALL PUT_AREG(U(A)+1, 64_REG(1){62,32});  
2221    36 * 11 CALL PUT_AREG(U(A), 64_REG(1){C31});  
2222    * 12 ENDIF  
2223    * 13 IF U(A)>U(B) *AND* (U(A)+1)<U(B), THEN  
2224    36 * 14 CALL PUT_AREG(U(B),SHIFT_COUNT);  
2225    * 15 ENDIF  
2226    * 16 CALL HALF_CFC_TOGGLE  
2227    * 17 RETURN  
*-----*
```

ANUVA-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 125

\_HCF ..COMPLEMENT A, FMT IV A, F=70 2

REF	PAGE	1	CALL GET_AREG([A], 32_REG[1])
2229	33	2	32_REG[1]; ** -NOT. 32_REG[1]
2230	*	3	CALL PUT_AREG([A], 32_REG[1])
2231	36	4	CALL HALFWORD_TOGGLE_E
2232	*	5	EE11EN
2233	*	*	

14 DEC 79 PAGE 121  
NSMC AN/UYK-7 (CP)  
Cp INSTRUCTION SET

\_HDCP == DOUBLE COMPLEMENT A, FMT IV A, F=70 3

REF PAGE \*\*\*\*\*  
2235 \* 1 CALL GET\_AREG(UA)+1, 32\_REG(21)  
2236 \* 2 32\_REG(21) =~ NOT\_ 32\_REG(21)  
2237 \* 3 CALL PUT\_AREG(UA)+1, 32\_REG(21)  
2238 \* 4 CALL \_HCP  
2239 \* 5 RETURN  
\*\*\*\*\*

NSWC ANUH-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 127

-HORIE\_HA,-HAN,-HICP,-HAND), FMT IV A, F=71 0 THPU 71 3 E F=71 5

REF  
PAGE \*\*\*\*\*  
2241 33 \* 1 CALL GET\_AREG(UA), 32\_REG(1)  
2242 33 \* 2 CALL GET\_AREG(U5), 32\_REG(2)  
2243 \* 3 DC CASE U(4);  
2244 \* 4 101 LOGICAL SUM  
2245 \* 5 32\_REG(1) := 32\_REG(1) + V. 32\_REG(2)  
2246 \* 6 \V SUM  
2247 \* 7 32\_REG(1) := 32\_REG(1) + 32\_REG(2) --ONES COMPLEMENT ADDITION.  
2248 \* 8 UPDATE FIXED POINT OVERFLOW INDICATOR(LASR3,1)  
2249 \* 9 \V DIFFERENCE  
2250 \* 10 32\_REG(1) := 32\_REG(1) - 32\_REG(2) --ONES COMPLEMENT SUBTRACTION.  
2251 \* 11 UPDATE FIXED POINT OVERFLOW INDICATOR(LASR3,1)  
2252 \* 12 \V LOGICAL DIFFERENCE  
2253 \* 13 32\_REG(1) := 32\_REG(1) .X. 32\_REG(2)  
2254 \* 14 \V AND  
2255 \* 15 32\_REG(1) := 32\_REG(1) .A. 32\_REG(2)  
2256 \* 16 ENDU -- ENDCASE  
2257 \* 17 CALL PUT\_AREG(UA), 32\_REG(1)  
2258 \* 18 CALL HALFWORD\_TOGGLE  
2259 \* 19 RETURN  
\*\*\*\*\*

\_HM \*MULTIPLY REGISTER. FMT 1W A, F=74 0

REF PAGE \*\*\*\*\*

```
2261   * 1   * A(UC(A)) X A(UC(B)) -> A(UC(A)+1); A(UC(A))
2262   33   * 2   CALL GET_AREG (UC(A)), 32_REG(2)  ..GET MULTIPLICAND.
2263   33   * 3   CALL GET_AREG (UC(B)), 32_REG(1)  ..GET MULTIPLIER.
2264   * 4   SIGN_INDICATOR != 32_REG(1)(31:1) *X* 32_REG(2)(31:1) ..REMEMBER SIGN FOR RESULT.
2265   96   * 5   CALL DO_MULTIPLY ..DO ACTUAL MULTIPLICATION
2266   * 6   JE SIGN_INDICATOR ==NEGATIVE", THEN ==NEGATE ANSWER.
2267   * 7   REPEAT_ACCUMULATOR ==NOT. REPEAT_ACCUMULATOR
2268   * 8   32_REG(1) == NOT. 32_REG(1)
2269   * 9   EB0E
2270   36   * 10  CALL PUT_AREG (UC(A)+1, REPEAT_ACCUMULATOR) ..STORE MOST SIGNIFICANT BITS
2271   36   * 11  CALL PUT_AREG (UC(A), 32_REG(1)) ..STORE LEAST SIGNIFICANT BITS
2272   36   * 12  CALL HALFWORD_TOGGLE
2273   * 13  RETURN
2274   * 14  *****
```

NSWC

N/UVK-7 (CP)  
CP INSTRUCTION SET

-MD ••DIVIDE REGISTER FMT IV A, F=74 1

P6F PAGE \*\*\*\*\*

```
2276   * 1  *(A(U(A+1)), A(U(A))) / A(U(C3)) -> A(U(C4)); REMAINDER -> A(U(A+1))
2277   * 2  CALL GET_AREG(U(0)), 32_REG(2); . . .GET DIVISOR.
2278   * 3  CALL GET_AREG(U(A)), 64_PEG(1){31|32}); . . .GET DIVIDEND.
2279   * 4  CALL GET_AREG(U(A)+1), 64_REG(1){63|32});
2280   * 5  CALL DO_DIVIDE {64_REG(1)}, 22_REG(2); . . .DO ACTUAL DIVISION.
2281   * 6  CALL PUT_AREG(U(A)+1), 64_REG(1){63|32}); . . .STORE REMAINDER.
2282   * 7  CALL PUT_AREG(U(A)), 64_PEG(1){31|32}); . . .STORE QUOTIENT.
2283   * 8  CALL HALFWORD_TOGGLE
2284   * 9  RETURN
```

\_HRT ..SQUARE ROOT FMT IV A F=74 2

REF PAGE \*\*\*\*\*

```

2286   * 1   ** THIS INSTRUCTION USES THE ALGORITHMS ON PAGE 445-452 OF THE UYK-7 LEARNER'S GUIDE
2286   * 2   ** (JANUARY 1973). UPON COMPLETION 32_REG(1) EQUALS THE SQUARE ROOT AND 64_REG(1) (63,32)
2286   * 3   CALL GET_AREG(1),61,64_REG(1)(31,32) **GET HIGH ORDER BITS OF RADICAND.
2290   * 4   IE 64_REG(1)(31,2) "UPPER 2 BITS OF RADICAND", THEN
2292   * 5   ASR(3,1) := 1 ..INDICATE OVERFLOW.
2293   * 6
2294   * 7   ELSE
2295   * 8   ASR(3,1) := 0 ..CLEAR OVERFLOW.
2296   * 9
2297   * 10  ENDIF
2297   * 11  64_PEG(1){63,32} := 0 ..INITIALIZE THE CURRENT RESIDUE.
2298   * 12  32_REG(1) := 0 ..INITIALIZE THE ROOT.
2298   * 13  CALL DO_SORT(64_REG(1), 32_REG(1)) ..WORK ON HIGH ORDER BITS OF THE RADICAND.
2299   * 14
2301   * 15  **NOTE: DO_SQRT USES 32_REG(2) FOR SCRATCH.
2301   * 16
2302   * 17  CALL GET_AREG(1), 64_PEG(1){31,32}) ..GET LEAST SIGNIFICANT BITS RADICAND.
2302   * 18  CALL DO_SORT(64_REG(1), 32_REG(1)) ..WORK ON LOW BITS OF RADICAND.
2304   * 19  CALL DO_SORT(64_REG(1), 32_REG(1)) ..COMPENSATE FOR EXTRA SHIFT IN DO_SQRT.
2305   * 20  32_REG(1) = 32_PEG(1) * RL(1) ..INITIALIZE THE ROOT.
2306   * 21  CALL PUT_AREG(1), 32_REG(1); ..STORE ROOT.
2306   * 22  CALL PUT_AREG(1), 64_REG(1){63,32}) ..STORE RESIDUE.
2307   * 23
2308   * 24  CALL HALFWORD_TOGGLE
2308   * 25
2309   * 26  RETURN
2309   * 27
*****
```

DC\_SORT (REF 64\_REG, REF 32\_REG)

REF PAGE \*\*\*\*\*  
2311 \* 1 COUNT = 16  
2312 \* 2 DO WHILE COUNT > 0  
2313 \* 3 64\_REG = 64\_REG .LL. 2 ..THE PARTIAL RADICAND (64\_REG(63,32)) IS FORMED BY SHIFTING THE  
2314 \* 4 CURRENT RESIDUE (64\_REG(63,32)) LEFT TWO BITS AND SUBSTITUTING TWO BITS FROM THE ORIGINAL  
2315 \* 5 ..RADICAND (64\_REG(31,32)). THE PARTIAL RADICAND (64\_REG(63,32)) MAY REQUIRE MORE THAN  
2316 \* 6 12 BITS TO EXPRESS.  
2317 \* 7 32\_REG = 32\_REG .LL. 1 ..SHIFT PARTIAL ROOT LEFT 1.  
2318 \* 8 32\_REG(2) := 32\_REG + 1 ..32\_REG(2) IS THE 32 BIT TRIAL ROOT EXTRACTOR.  
2319 \* 9 IF 32\_REG >= 32\_REG(2), THEN ..UNSIGNED 34 BIT COMPARE (DO BE SAFE).  
2320 \* 10 32\_REG := 32\_REG - 32\_REG(2) ..32\_REG NOW HAS THE NEW CURRENT RESIDUE.  
2321 \* 11 32\_REG(1,1) := 1 ..SET BIT IN ROOT.  
2322 \* 12 ENDIF  
2323 \* 13 COUNT := COUNT-1  
2324 \* 14 ENDDO  
2325 \* 15 RETURN  
\*\*\*\*\*

AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 132

\_HIB ::LOAD B(A) WITH B(B) FMT IV A, F=74 3

REF  
PAGE

```
*****  
*   * 1    --B(U(B)) -> B(U(A))  
*   * 2    CALL GET_REG(U(B), 32_REG(1))  
*   * 3    CALL GET_BREG(U(A), 32_REG(2))  
*   * 4    32_REG(2)(15) = 32_REG(1)(15)  
*   * 5    CALL PUT_BREG(U(A), 32_REG(2))  
*   * 6    CALL HALFWORD_TOGGLE  
*   * 7    RETURN  
*****
```

MSAC ANNUAL REPORT (CP) CP INSTITUTE OF EDUCATION SETI

14 DEC 79 PAGE 133

HC ::COMPARE REGISTER:: ENT IX A: F=74 4

REF PAGE	*	1	• A(U[0]) IS COMPARED TO A(U[0]) ; SET ASR{2,2} "CD" ACCORDINGLY.
2338	*	1	CALL GET_AREG(U[0]), 32_REG(1))
2339	*	2	CALL GET_AREG(U[0]), 32_REG(1))
2340	*	3	CALL GET_AREG(U[0]), 32_REG(2))
2341	*	4	CALL SET_CD(32_REG(1)), 32_REG(2))
2342	*	5	CALL HALFWORD_TOGGLE
2343	*	6	RETURN

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 134

\_HCL \*\*COMPARE LIMITS, REGISTER FMT IV A, F=74 5

REF	PAGE	*****
2345	*	** A((A)+1) > A((B)) > A((A)) -> SET ASP(C,1) -> OUTSIDE/WITHIN LIMITS.
2347	33 *	CALL GET_AREG((A), 32_REG(2))
2348	33 *	CALL GET_AREG((A)+1, 32_REG(3))
2349	33 *	CALL GET_AREG((B), 32_REG(1))
2350	30 *	CALL SET_CD2(32_REG(1)), 32_REG(2), 32_REG(3))
2351	*	CALL HALFORD_TOGGLE
2352	*	RETURN
	*	*****

NSWC

AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 135

-HCR --COMPARE MASKED, REGISTER FMT IV A, F=74, 6

REF

PAGE \*\*\*\*\*

2354 \* 1 .. A((A)+1) -A. A((A)) IS COMPARED TO A(U{B}) ; SET ASR(z,2) "CD" ACCORDINGLY.  
2355 \* 2 CALL GET\_AREG(A), 32\_REG(1)  
2356 \* 3 CALL GET\_AREG(U(A)+1), 32\_REG(2)  
2357 \* 4 32\_REG(1) = 32\_REG(1) .AND. 32\_REG(2)  
2358 \* 5 CALL GET\_AREG(U(B)), 32\_REG(2)  
2359 \* 6 CALL SET\_CCL(32\_REG(1), 32\_REG(2))  
2360 \* 7 CALL HALFWRG\_TOGGLE  
2361 \* 8 RETURN  
2362 \* 9 \*\*\*\*\*

NSWC INVUK-7 (CP:  
CP INSTRUCTION SET

14 DEC 79 PAGE 136

-HCB --COMPARE B(U(8)) WITH B(U(A)) FMT IV A F=74 7

REF  
PAGE

2364	*	1	** B(U(8))(15,16) IS COMPARED TO B(U(A))(15,16) ; SET ASR(2,2) "CD" ACCORDINGLY
2366	34	2	CALL GET_BREG(U(8), 32_REG(1))
2367	34	3	CALL GET_BREG(U(A), 32_REG(2))
2368	*	4	32_REG(1) = 32_REG(1)(15)
2369	*	5	32_REG(2) = 32_REG(2)(15)
2370	29	6	CALL SET_CD(32_REG(1), 32_REG(2))
2371	*	7	CALL HALFWORD_TOGGLE
2372	*	8	RETURN

AN/UYK-7 (CCP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 137

- HSIM --STAGE ICC MONITOR CLOCK IN A FPT IN A C=77 0

REF PAGE \*\*\*\*\*  
2374 \* 1 IF ASPLIS,6)=0 "TASK MODE", THEN ..PRIVILEGED INSTRUCTION.  
2375 \* 2 CALL GENERATE\_SYNCHRONOUS\_INTERRUPT (\*P=-1), PRIVILEGED INSTRUCTION!  
2377 \* 3 ..ABORT THE INSTRUCTION.  
2378 \* 4 ENDIE  
2379 \* 5 CALL CP11CC\_CLOCK\_COMMUNICATIONS  
2380 \* 6 GET Q BUS INTG 32\_REG12  
2381 \* 7 CALL PUT\_AREG1U(0), 32\_REG12)  
2382 \* 8 CALL HALFWORD\_TOGGLE  
2383 \* 9 BEIUES  
\*\*\*\*\*

NSW-

AP/UYK-7 (CP)  
CP-INSTRUCTION SET

-HSTC --STORE REAL-TIME CLOCK IN A FPT IN A F-77 1

REF  
PAGE

2385	22	*	1	CALL CP/IOC_CLOCK_COMMUNICATIONS
2386	*	2	*	GET Q_BUS INTO 32_REG(12)
2387	36	*	3	CALL PUT_ARGUMENT, 32_REG(12)
2388	*	4	*	CALL HALFWORD_TOGGLE
2389	*	5	*	RETURNS

14 DEC 79 PAGE 138

NSC  
AN/URK-7 (CP)  
C2 INSTRUCTION SET

16 DEC 79 PAGE 139

\_HPI(E\_HAI) ==>PREVEI(I(ALLO)) CLASS III INT. FMT IV A, F=77 & E F=77 5

REF

PAGE

```
*****  
* 1 AE ASR(19,4)=0 "TASK MODE", THEN **PRIVILEGED INSTRUCTION.  
* 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1", "PRIVILEGED INSTRUCTION VIOLATION")  
* 3 .ABORT THE INSTRUCTION.  
* 4  
* 5 AE LCF(4) = 4 "HPI", T=EN  
* 6 ASR(12,1) := 1 ..LOCKOUT CLASS III INTERRUPTS.  
* 7 ELSE .."HAI"  
* 8 ASR(12,1) := 3 ..ENABLE CLASS III INTERRUPTS.  
* 9 ENDIF  
* 10 CALL HALFWORD_TOGGLE  
* 11 RETURN  
*  
*****
```

NSMC AN/UYK-7 (CP)  
CP INSTRUCTION SET

14 DEC 79 PAGE 140

-H776 ..HALT AND WAIT AND WAIT FOR INTERRUPT FRT IV A F-77 6

REF

PAGE

```
*****  
2404   * 1 IF ASPLG4=0 "TASK MODE", THEN --PRIVILEGED INSTRUCTION.  
2405   * 2 CALL GENERATE_SYNCHRONOUS_INTERRUPT("P-1", PRIVILEGED INSTRUCTION VIOLATION)  
2406   * 3 --ABORT THE INSTRUCTION.  
2407   * 4 ENDIE  
2408   * 5  
2409   * 6 IF .NOT. U(I), THEN --HALT INSTRUCTION.  
2410   * 7 STOP PROCESSING INSTRUCTIONS WITH STOP & INDICATOR SET  
2411   * 8 ELSE --HALT AND WAIT FOR INTERRUPT.  
2412   * 9 STOP PROCESSING INSTRUCTIONS  
2413   * 10 WAIT FOR AN ASYNCHRONOUS OR CP MONITOR CLOCK INTERRUPT.  
2414   * 11 UPON INTERRUPT PROCESS IT AND CONTINUE WITH NEXT INSTRUCTION  
2415   * 12 CALL HALFWORD_TOGGLE  
2416   * 13 RETIEN  
2417   *  
*****
```

NSWC

AN/UYK-7 (CP)

16 DEC 79 PAGE 141

\*  
\* AN/UYK-7 CP EMULATION GLOSSARY \*  
\*  
\*\*\*\*\*

## AN/UYK-7 CP EMULATION STATUS/CONTROL INDICATORS

- \*\*\*\*\*  
2420     \* 1     TO SUPPORT REENTRANCY ALL GLOBAL CONTROL VARIABLES ARE CONSIDERED  
2421     \* 2     TO BE ACCESSIBLE BY ALL PROCEDURES BUT CONTAINED WITHIN A WORK SPACE  
2422     \* 3     FOR A PARTICULAR EMULATOR.
- 2423     \* 4     CHARACTER\_ADDRESSING\_OVERRIDE - WHEN SET, CHARACTER ADDRESSING IS  
2424     \* 5     IN EFFECT. USED TO ACQUIRE CORRECT  
2425     \* 6     CHARACTER SPECIFICATION VALUES (I.E.  
2426     \* 7     -C, -P, AND -MASK) WHEN REPEATING  
2427     \* 8     INSTRUCTIONS USING CHARACTER ADDRESSING.
- 2428     \* 9     CP\_MONITOR\_CLOCK\_INTERRUPT - WHEN SET, THE CP MONITOR CLOCK HAS GONE  
2429     \* 10    NEGATIVE AND THE ASSOCIATED CLASS II  
2430     \* 11    INTERRUPT IS HELD PENDING UNTIL PROCESSED.
- 2431     \* 12    EXECUTE\_REMOTE\_IN\_PROGRESS - WHEN SET, AN INSTRUCTION IS BEING EXECUTED  
2432     \* 13    VIA THE XP (F-02 2) OR XPL (F-02 3)  
2433     \* 14    INSTRUCTION. USED TO SUPPRESS CHANGING OF  
2434     \* 15    THE ASP UPPER/LOWER HALF-WORD BIT WHEN A  
2435     \* 16    HALF-OPC INSTRUCTION IS EXECUTED VIA THE  
2436     \* 17    XP OR XPL INSTRUCTION.
- 2437     \* 18    INSTRUCTION\_FORMAT\_INDICATOR - THIS INDICATOR CONTAINS THE FORMAT  
2438     \* 19    (I.E. I, II, III, IV) OF THE INSTRUCTION  
2439     \* 20    THAT IS CURRENTLY EXECUTING.
- 2440     \* 21    INTERRUPT\_INTERRUPT - WHEN SET, AN INTERPROCESSOR INTERRUPT IS  
2441     \* 22    PENDING. THIS DATA STRUCTURE MUST RECEIVE  
2442     \* 23    THE CP ID OF THE PROCESSOR CAUSING THE  
2443     \* 24    INTERRUPT.
- 2444     \* 25    INTERRUPT\_SCAN\_INHIBIT! - WHEN SET (I.E. BETWEEN EXECUTION OF  
2445     \* 26    TWO HALF-WORD INSTRUCTIONS AND DURING  
2446     \* 27    REPEATS OF CP REFERENCE INSTRUCTIONS  
2447     \* 28    (F=54-57), THE ASYNCHRONOUS INTERRUPT  
2448     \* 29    SCAN IS NOT PERFORMED. FURTHERMORE, THE  
2449     \* 30    MAIN LOOP LOGIC SUPPRESSES INTERRUPT  
2450     \* 31    SCANNING WHEN IN INTERRUPT MODE.  
2451     \* 32    \*\*\*\*\*

## STATUS/CONTROL INDICATORS PAGE 2

2453	+ 1	<b>MEMORY_STORE_INDICATOR</b>	- THIS INDICATOR IS SET WHEN AN INSTRUCTION
2454	+ 2		- STOOPS A QUANTITY INTO MEMORY. THIS FACT IS REQUIRED FOR REPEAT TERMINATION LOGIC.
2455	+ 3		- WHEN SET, ALL BREAKPOINT REGISTERS WILL
2456	+ 4		- BE ACTIVE (TOTAL OF 8), ELSE ONLY THE
2457	+ 5		- EMULATED HARDWARE REGISTER WILL BE ACTIVE.
2458	+ 6		NOTE, THE UYK-7 EMULATOR HAS EIGHT (8)
2459	+ 7		BREAKPOINT REGISTERS. ONE REGISTER
2460	+ 8		CORESPONDS TO THE ACTUAL HARDWARE
2461	+ 9		BREAKPOINT REGISTER OF A REAL UYK-7 MACHINE.
2462	+ 10		THE OTHER SEVEN (7) BREAKPOINT REGISTERS
2463	+ 11		ARE CALLED PSEUDO BREAKPOINT REGISTERS AND
2464	+ 12		ARE CONSIDERED EXTENSIONS TO THE UYK-7
2465	+ 13		ARCHITECTURE. PSEUDO BREAKPOINTS ARE NOT
2466	+ 14		ACCESSIBLE TO UYK-7 PROGRAMS.
2467	+ 15		- WHEN SET, INSTRUCTION IS BEING REPEATED.
2468	+ 16	<b>REPEAT_IN_PROGRESS</b>	- SET BY EXECUTION OF A REPEAT INSTRUCTION.
2469	+ 17	<b>REPEAT_PENDING</b>	- REPEAT IS NOT IN PROGRESS UNTIL THE PENDING
2470	+ 18		INSTRUCTION HAS BEEN Fetched FROM MEMORY.
2471	+ 19		- WHEN SET, SOFTWARE PROTECTION REGISTER
2472	+ 20	<b>SPR_CHECKS</b>	- CHECKS ARE ENABLED. SPR CHECKS ARE
2473	+ 21		DISABLED WHEN CLASS II INTERRUPTS ARE
2474	+ 22		LOCKED OUT. INTERRUPT BASE REGISTERS ARE
2475	+ 23		SELECTED OR MEMORY LOCKOUT INHIBIT IS SET.
2476	+ 24		- WHEN SET, ONE OF THE CONDITIONS CAUSING
2477	+ 25	<b>SPR_PRIVILEGED_INSTRUCTION</b>	USE OF INDIRECT ADDRESSING WITH SPR(16,1)
2478	+ 26		TO BE A PRIVILEGED OPERATION IS IN EFFECT.
2479	+ 27		THESE CONDITIONS INCLUDE:
2480	+ 28		(1) CERTAIN DOUBLE-WORD INSTRUCTIONS
2481	+ 29		(ALL EXCEPT OS 4).
2482	+ 30		(2) ALL REPEATED INSTRUCTIONS.
2483	+ 31		(3) U(F) = 25 OR U(F) = 11.
2484	+ 32		(4) ALL FORMAT III INSTRUCTIONS.
2485	+ 33		

## AN/UYK-7 CP ARCHITECTURE DESCRIPTORS

2467	+ 1	BREAKPOINT_REGISTER	- CMR ADDRESS OF UYK-7 HARDWARE BREAKPOINT REGISTER (60).
2488	+ 2	-CMR	- (CONTROL) MEMORY REGISTERS SECTION OF CP WHERE THE ADDRESSABLE REGISTERS ARE LOCATED.
2489	+ 3		INCLUDED ARE TASK AND INTERRUPT ACCUMULATORS, INDEX(8), AND BASE(8) REGISTERS. SP005 AND Z, R05, THE A\$, CP MONITOR CLOCK AND BREAKPOINT REGISTER(L) AND ICW AND USW FOR EACH INTERRUPT LEVEL.
2490	+ 4		CMR LOCATION 10 IS UNASSIGNED BIT ADDRESSABLE BY PROGRAMS.
2491	+ 5		- MAIN MEMORY SUPPLYING 32-BIT WORD STORAGE.
2492	+ 6		- AMOUNT OF 32-BIT MAIN MEMORY (UP TO 256 K) AVAILABLE FOR USE BY CP
2493	+ 7		- 512 WORD NON-DESTRUCTIVE READ OUT MEMORY. USUALLY CONTAINS DIAGNOSTIC, BOOTSTRAP, RECOVERY AND INTERRUPT SWITCHES.
2494	+ 8		PROGRAMS.
2495	+ 9		
2496	+ 10		
2497	+ 11	_MAIN	
2498	+ 12		
2499	+ 13	MEMCFLIMIT	
2500	+ 14		
2501	+ 15	_NDRC	
2502	+ 16		
2503	+ 17		
2504	+ 18		
2505	+ 19		

AIA/U7C-7 CP PAINTERMANCE PANEL SUPPORT

- 2507      1      **BUSSTRAP\_SWITCH**
- + 2
- + 3      **-STEP**
- + 4
- + 5
- + 6
- + 7
- + 8      **STOP\_SWITCH**
- + 9
- + 10
- + 11
- + 12
- + 13
- 2508     2
- 2509     3
- 2510     4
- 2511     5
- 2512     6
- 2513     7
- 2514     8
- 2515     9
- 2516     10
- 2517     11
- 2518     12
- 2519     13
- A 3-POSITION SWITCH (0,1,2) WHICH  
SELECTS AN MBC STARTING ADDRESS.
- WHEN SET, THE EMULATOR SUSPENDS OPERATION  
AT THE BEGINNING OF THE MAIN LOOP UNTIL  
RESTARTED. THIS CORRESPONDS TO THE MODE  
SWITCH (I.E. SET -> INSTRUCTION MODE  
AND CLEAR -> RUN MODE).
- SERIES OF THREE SWITCHES (NUMBERED  
5,6,7) FOUND ON THE MAINTENANCE PANEL  
WHICH ALLOW STOPPING UNDER PROGRAM CONTROL.
- SERIES OF THREE SWITCHES (NUMBERED  
1,2,3) FOUND ON THE MAINTENANCE PANEL  
WHICH ALLOW BRANCHING UNDER PROGRAM CONTROL.

## AN/UYK-7 CP EMULATION INTERNAL REGISTER SUPPORT

2521	1	-C	- THE C DESIGNATOR FOR CHARACTER ADDRESSING
2522	2	-	- C FIELD OF LAST EXECUTED IAR.
2523	3	N	- THE MEMORY REFERENCE REGISTER. A 32 BIT REGISTER CONTAINING THE ADDRESS OF THE LAST MEMORY LOCATION REFERENCED.
2524	4		- THE CHARACTER MASK NEEDED TO ACQUIRE
2525	5		- 4 BITS FOR CHARACTER ADDRESSING.
2526	6	-MASK	- THE PROGRAM COUNTER REGISTER. A 20-BIT REGISTER WITH LOW ORDER 16 BITS (P10) FORMING A DISPLACEMENT THAT IS ADDED TO THE BASE REGISTER INDICATED BY BITS 19 THROUGH 17.
2527	7	P	BIT 16 IS UNUSED.
2528	8		- LEAST SIGNIFICANT BIT ADDRESS FOR CHARACTER ADDRESSING.
2529	9		- CONTAINS THE A & B FIELDS OF THE LAST REPEAT INSTRUCTION AS USED IN REPEAT TERMINATION LOGIC.
2530	10		- A 32-BIT WORKING REGISTER WHICH HOLDS A COPY OF THE APPROPRIATE ACCUMULATOR TO TESTED BY REPEAT TERMINATION LOGIC. IT IS THE RESPONSIBILITY OF EACH INSTRUCTION (WHERE APPLICABLE) TO ENSURE THE PROPER CONTENTS OF THIS REGISTER.
2531	11		- CONTAINS THE SY MODIFIER OF THE LAST REPEAT INSTRUCTION AS USED IN REPEAT TERMINATION LOGIC.
2532	12		- THE INSTRUCTION REGISTER. A 32-BIT REGISTER WHICH RECEIVES THE INSTRUCTION WORD FROM MEMORY AND PROVIDES THE CONTROL NECESSARY TO START EXECUTION OF THE INSTRUCTION FIELDS (E.G. F1, F2, A, ETC.) ARE AS PER REPERTOIRE CARB.
2533	13		- LOWER 16 BITS OF U.
2534	14	-P	- UPPER 16 BITS OF U.
2535	15		
2536	16	REPEAT_AB	
2537	17		
2538	18		
2539	19		
2540	20		
2541	21		
2542	22		
2543	23		
2544	24		
2545	25	REPEAT_SY	
2546	26		
2547	27		
2548	28	U	
2549	29		
2550	30		
2551	31		
2552	32		
2553	33		
2554	34	UL	
2555	35	JU	
2556	36		
2557	37		NOTE: 32_FEF IS A GENERIC TERM FOR A 32-BIT REGISTER. THIS DESIGN ASSURES 4 SUCH WORKING REGISTERS WHICH ARE REFERENCED AS 32_REG(1), ..., 32_REG(4). THESE REGISTERS DO NOT NECESSARILY CORRESPOND TO ANY SPECIFIC UYK-7 INTERNAL REGISTERS.
2558	38		REFERENCE PARAMETERS ARE INDICATED BY THE 'REF' CONSTRUCT IN THE SUBPROCEDURE DECLARATION. CHANGING A REFERENCE PARAMETER MODIFIES THE ARGUMENT IN THE CALLING ROUTINE.
2559	39		
2560	40		
2561	41		
2562	42		
2563	43		
2564	44		

NSWC

AN/UYSK-7 (CP)

14 DEC 79 PAGE 147

\*\*\*\*\*  
\* REFERENCES \*  
\*\*\*\*\*

## REFERENCES

- + 1. CAINE, FARBER, AND GORDON, "PROGRAM DESIGN LANGUAGE REFERENCE GUIDE", JULY 1975.
- + 2. NAVSHIPS 0967-051-6291, AN/UYK-7(V) COMPUTER SET, MARCH 1977.
- + 3. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYK-7 COMPUTER MAINTENANCE COURSE VOLUMES 1, 2 & 3, JANUARY 1977.
- + 4. NAVSHIPS 0967-319-4012, TECHNICAL MANUAL FOR COMPUTER SET AN/UYK-7(V) VOLUMES 1 & 2, JANUARY 1974.
- + 5. SPERRY UNIVAC, STUDY GUIDE FOR AN/UYK-7 COMPUTER, VOLUME 1, MARCH 1973.

NSMC

AN/UYK-7 (CP)

14 DEC 79 PAGE 149

\*\*\*\*\*  
\* INDEX TO DATA ITEMS \*  
\*\*\*\*\*  
\*\*\*\*\*

INDEX TO DATA ITEMS

PAGE LINE TYPE

NAME AND REFERENCES			
DI	ABS_ADDR	17	BPR_CHECK 11 17 28
DI	ACCUMULATOR_SIGN	96	DC_MULTIPLY 25 29
DI	AUTO_RECEIVER	8	INTERRUPT_SEQUENCE 19
DI	A_DESIGNATOR	2E	UPDATEA_REPLACE 1 4 6
	33	GET_AREG	2 6 6 8 10
	36	PUT_AREG	1 2 5 5 7 9
DI	A_MANTISSA	4D	FLOATING_ADD_SUBTRACT_HEADER 2 9 10 14 23
DI	BCDSTRAP_SWITCH	8	INTERRUPT_SEQUENCE 21
DI	BREAKPOINT_REGISTER	17	BPR_CHECK 25 26 27 28
DI	B_DESIGNATOR	34	GET_BREG 2 6 6 7 11 13
	37	PUT_BREG	2 6 6 7 9 11
DI	CALL_REPLACE	92	_BZ 9
DI	CHARACTER_ADDRESSING_OVERRIDE	4	I_SEQUENCE 17

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES		
01			CHARACTER_ADDRESSING_OVERRIDE		
	14	DP_READ		12	
	15	NP_STORE		12	
	18	IA_SEQUENCE		12	
	38				
71		CHECK_TYPE	20	SPR_CHECK	9 14 15
31		CP_MONITOR_CLOCK_INTERRUPT	5	INTERRUPT_SCAN	10 13
01		DUR_C	13	JUMP_ADDRESS	7
01		DUR_C1	13	JUMP_ADDRESS	7
01		DUR_MASK	13	JUMP_ADDRESS	7
01		DUR_P	13	JUMP_ADDRESS	7
31		EXECUTE_REMOTE_IN_PROGRESS	4	I_SEQUENCE	16
	25	HALFWORD_TOGGLE			
	55	_XR		5	
					17
01		FLOATING_ADD_SIZE_HDR	65	_FA	1
	66	_FAN			1

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI	DI	FLUTATINI_POINTS-END	
	68	-FO	48
DI	DI	GENERATE_SYNCHROBOS	
	119	-HSC_60	6
	120	-HLC_61	6
DI	DI	HALFWORD_TOGGLE	
	119	-HSC_60	9
	120	-HLC_61	9
	121	-HLC	16
	122	-HDLC	12
	123	-HSF	15
	124	-HDHF	16
	125	-HEP	4
	127	-HGR	18
	128	-HM	12
	129	-HD	6
	130	-HPT	18
	132	-HLB	6
	133	-HC	5
	134	-HCL	5
	135	-HCM	6
	136	-HCB	7
	137	-HSIM	7
	138	-HSTC	6

NSWC ANALYST-7 (CP)  
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	136		_HPI 10
	140		_HT75 12
SI	INSTRUCTION_FORMAT_INDICATOR 4_I_SEQUENCE 20		
SI	INSTRUCTION_FORMAT_INDICATOR 14_OP_READ 16 30		
	15 GP_STORE 26		
	55 _AF 16		
SI	INTERPROCESSOR_INTERRUPT 5_INTERRUPT_SCAN		
	19 22		
SI	INTERRUPT_SCAN_INITIAT 3 CP_MAIN_LOOP 17 25 4 I_SEQUENCE 26 6 REPEAT_SEQUENCE 32		
SI	IICC_		
	12 GET_ISC 6 7 1G 12		
SI	IICC_CLOCK_COMMUNICATIONS 137_HSTM 5		
	138 _HSTC 1		
SI	JUMP_STOP 115 _J532 3 4 9 26 28		
	116 J533 4 5 1G 22 24		
SI	JUMP_S_ITC 115 _J532 14		

NSML AN/DYK-7 (CP)  
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.005

PAGE	LINE	TYPE	NAME AND REFERENCES
	116	J533	15
DI	MAIN_MEMORY	19	MEMORY_READ 9 14
GI	MB_CARRY	56	DO_MULTIPLY 4 6
GI	MEMORY_STORE_INDICATOR	3	CP MAIN LOOP 13
	9	REPEAT_SEQUENCE	
	15	OP_STORE	25 26
	15		15
DI	M_DISPLACEMENT	13	JUMP_ADDRESS 3 9 11
	14	OP_READ	
	15	OP_STORE	4 7 10 10
DI	M_MANTISSA	40	FLOATING_ADD_SUBTRACT_HEADER 1 6 7 11 19
DI	OPERAND_DISPLACEMENT	13	JUMP_ADDRESS 5 7 9 9 11 14 15
	14	OP_READ	
	15	OP_STORE	7 7
	55	-XR	7 7
	105	-JEP	3 6 9
	106	-DJZ	2
	107	-DJNZ	2 7
	108	-FS1	2 13

NSRC AN/UYK-7 (CP)  
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149-006

INDEX TO DATA ITEMS

PAGE LINE TYPE

		NAME AND REFERENCES	
D1	OPERAND_S	13	JUMP_ADDRESS
109	-LBJ	7	12 15 16 17
110	-JBNZ	8	10
112	-JL	6	6
113	-J530	4	7
114	-J531	2	7
115	-J532	3	32
116	J533	24	27
		2	23
55	-XR	3	7
105	-JEP	2	16
106	-JJZ	2	7
107	-DJNZ	2	7
108	-FS1	2	7
109	-LBJ	2	13
110	-JBNZ	8	10
112	-JL	6	6
113	-J530	4	6
114	-J531	2	7
115	-J532	3	32
116	J533	24	27
		2	23
OPERAND_DISPACEMENT		1C5 -JEP 16	

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI	0_BUS	12	GET_ISC 8 10
		22	CPLIC_CLOCK_COMMUNICATIONS 8 9 10 11 12 13
	71	-AFI	9 11 11 12 13
	72	-LIM	9 11 11 12 13
	73	-IO	9 11 11 12 13
	137	-MSIM	14 15 16
	138	-MSTC	6
		2	
DI	PSEUDO_BREAKPOINTS	17	BPR_CHECK 7
DI	PSEUDO_BREAKPOINTS	17	BPR_CHECK 19
DI	F_MOD	19	MEMORY_READ 5 11
DI	P_MODIFICATION	11	GENERATE_SYNCHRONOUS_INTERRUPT 4 44
	20	SPR_CHECK	10 18 24 30 34
DI	REPEAT_AB	9	REPEAT_SEQUENCE 8 19
	27	-REPLACE	5
	75	-RP	13
DI	REPEAT_ACCUMULATOR	9	REPEAT_SEQUENCE 20 21 22 23 25 26
	48	-JP	2 3 3 4

NS\*C ANU K-7 (CP)  
INDEX 1] DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	49	-SC	3 4 4 5
	50	-MS	3 5 5 6
	51	-XDR	3 5 5 6
	52	-ALP	2 3 3 4
	53	-LP	1 4 4 6
	54	-CNT	3 5 5 8 12 14
	55	-SLP	2 6 6 10
	56	-SLP	1 3 3 4
	57	-SSUM	1 3 3 9 10
	58	-SGIF	2 ~3 3 9 10
	6G	-TSF	4 5 10 11
	67	-FM	
	76	-LA	14 15 16 17 21 24 26
	77	-LKB	1 2
	78	-LDIF	2 3
	79	-LNA	1 3 3 5
	80	-AA	2 3 3 5
	81	-LSUM	2 3 3 5
	82	-LNA	2 3 3 5
	83	-LM	1 2 3 3 5
	85	-AB	4 6 7
	86	-ANB	3 4 6 7
	87	-SB	2 3 3 4 ~
	88	-SA	2 3
	90	-SNA	2 3 3 ~

NSwC

ANFUK-7 (CP)  
INDEX TO DATA ITEMS

## INDEX TO DATA ITEMS

## PAGE LINE TYPE

## NAME AND REFERENCES

	91	_SN	1	2	3	3	5
	92	_BZ	3	5	7	9	
	93	_RA	1	4	6	6	9
	94	_PI	1	3	5	5	6
	95	_M					
	96	DO_MULTIPLY	7	7	9		
		1	4	9	12	13	16
		25	26	27	28	30	33
		45	48	49	50	50	53
	117	LCT	11	12			
	118	SCT	10	10	12		
	128	_HM	7	7	10		
DI	REPEAT_IN_PROGRESS						
	3	CP_MAIN_LOOP					
		12	21				
	4	I_SEQUENCE					
		23					
	5	INTERRUPT_SCAN					
		28	33				
	9	REPEAT_SEQUENCE					
		5	10	11	12	13	15
		26	31				
	11	GENERATE_SYNCHRONOUS_INTERRUPT					
		42					
	18	IA_SEQUENCE					
		29					
	27	_REPLACE					
		5					
	117	LCT					
	118	SCT	6	13			
			6	13			
DI	REPEAT_PENDING						
	4	I_SEQUENCE					
		21	24				
	5	INTERRUPT_SCAN					
		33					
	11	GENERATE_SYNCHRONOUS_INTERRUPT					
		41					

NSWC

AN/UTK-7 (CP)  
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
			75    -RP    12
DI	REPEAT_SY	9	REPEAT_SEQUENCE 3
			75    -RP    14
DI	RESUME_TYPE	19	MEMORY_READ 5    11
DI	REUND_BIT	68	-FD    31    33
DI	SHIFT64_REG	122	-HDLC 6
DI	SHIFT_COUNT	24	GET_SHIFT_AMOUNT 3    5    9    12
			40    FLOATING_AGO_SUBTRACT_HEADER 17    16    19    21    22    23
			42    FLOATING_NORMALIZE 4    6    8    9    10
			121    -HLC 122    -HDLC    2    6    7    8
			123    -HSF    3    5    7    8    9    13
			124    -HDSF    4    6    8    9    14
DI	SIGN_DIVIDEND	97	DU_DEVICE 10    26
DI	SIGN_IND	68	-FD 6    43
		97	DU_DIVIDE 5    23

NSWC AN/LYK-7 (CP)  
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.011

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
01		SIGN_INDICATOR	
	67	-FM	6 23
	95	-H	4 6
	128	-HM	4 6
01		SPR_CHECKS	
	3	CP MAIN LOOP	
	20	SPR_CHECK	8 10
	12		
01		SPR_PRIVILEGED_INSTRUCTION	
	4	I_SEQUENCE	18 25
	20	SPR_CHECK	16
	40	FLOATING_ADD_MINUS_HEADER	6
	59	-DS	1
	61	-DL	1
	62	DA	2
	63	-DC	3
	67	-FM	1
	68	-FD	1
	77	-LXB	1
	89	-SX8	2
	105	-JEP	1
	106	-DJZ	1
	107	-DJNZ	1
	109	-FSI	1
	106	-LEJ	1

NSWC AN/UYK-7 (CP)  
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	110		-JBNZ 1
	112		-JL 1
	113		-JS30 1
	114		-JS31 1
	115		-JS32 2
	116		-JS33 1
DI		STEP_SWITCH	
	115		-JS32
	116		JS33 1C
			11
DI		S_DESIGNATOR	
	14		OP_READ 6
	15		DP_STORE 6
	16		IA_SEQUENCE 11
	19		MEMORY_READ 2
			7
	23		DC_JUMP 4
	32		ADD_S 3
	35		GET_SREG 2
	36		PUT_SREG 1
	73		-IE 6
			12
DI		S_INITIALIZE	
	25		SDB_CHECK 13
			13
DI		TYPE-	
	17		SPP_C-ECK 13
	26		ZT 31 38

NSWC AN/UYK-7 (CP)  
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149-013

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI	UPDATE_REPLACE		93 _RA 94 _RI 8
DI	16_REG	20	SPR_CHECK
DI	18_REG	19	MEMORY_READ 2 7
DI	32_REG	4	1_SEQUENCE 3 6 13 JUMP_ADDRESS 14 16 17 18 14 OP_READ 15 OP_STORE 16 IA_SEQUENCE 10 11 12 13 19 MEMORY_READ 3 8 14 24 GET_SHIFT_AMOUNT 8 9 11 12 26 REPLACE_CHECK 3 5 27 REPLACE 1 4 8 11 11 28 UPDATE_REPLACE 3 6 7 32 ADD_S 8 9 33 GET_AREG 1 4 8 10 34 GET_BREG 1 4 8 11 13 35 GET_SREG 1 3 7 9 36 PUT_AREG 1 4 7 9 37 PUT_BREG 1 4 9 11

INDEX TO DATA ITEMS

PAGE LINE TYPE NAME AND REFERENCES

NSWC AN/UYK-7 (CP)  
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149-015

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES			
	69	-XS	34	37	39	46
	70	-IPI	4	5		
	71	-AEI	6	9	10	11 15
	72	-LIP	8	9		
	73	-IG	8	9	9	11
	77	-LXB	9	10		
	78	-LDIF	4	5	5	6
	79	-ANA	1	3		
	80	-AA	1	3		
	81	-LSUM	1	3		
	84	-LB	2	3	4	5
	85	-AB	2	3	4	5
	86	-ANB	2	4	5	5
	89	-SAB	4	5	5	6
	92	-BZ	1	5	7	10
	93	-RA	2	4	6	
	95	-R	2	3	4	7
	96	DD_MULTIPLY	1	1	3	6
	97	DD_DIVIDE	21	26	27	33
	98	-D	49	49	36	36
	99	-BC	2	5	44	44
	100	-CXI	3	4	46	46
	101	-C	2	3	4	4

## INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
	102	-CL	3 4 5 5 5
	103	-CM	3 4 5 5 6
	104	CG	3 4 4 5 6
	105	-JEP	5 6 7 7 11
	109	-LBJ	5 6
	110	-JBNZ	4
	111	-JS	1 2 3 4 5 6
	112	-JL	8 10 11 13 14
	115	-J532	21 22 23
	117	LCT	4 6 12 14 16
	118	SCT	2 4 6 10 14 16
	119	HSC-60	2 4 6 10 14 16
	120	HLC-11	14 17 20 22 27
	120	HLC-61	
	121	HLC-13	15 18 21 25
	121	-HLC	3 5 6 7 9
	123	-HSF	1 2 3 5 9 19
	125	-HCP	1 2 2 3
	126	-HDACP	1 2 2 3
	127	-HCR	1 2 5 5 7 7 10 10 13 13
	128	-HM	15 15 17 5 7 7 10 10 13 13
	129	-HD	2 3 4 4 8 8 11
	130	-HRT	2 5
	131	DG_SORT	2 10 11 12 14 15 15 16
	132	-HLE	7 8 8 9 9 10 10 10 10 11
	133	-HC	2 3 4 4 5

**NSWC ANNUAL INDEX TO DATA ITEMS  
INDEX TO DATA ITEMS  
ANNUAL (CP)**

#### **INDEX TO DATA ITEMS**

PAGE	LINE	TYPE	NAME AND PREFERENCES		
			134	-HCL	2
			135	-HCM	2
			136	-HCB	2
			137	-HSIP	2
			138	-HSTC	6
					2
DI	64_REG	41	FLOATIN	1	
		42	FLOATIN	1	
		43	FLOATIN	1	
		65	DEVICE_	1	
				1	
		62	DA	1	
		63	-DC	1	
		65	-FA	1	
		66	-FAN	1	
		68	-FD	1	
				5	
		97	...DO_DIVI	27	
				1	
		98	-D	27	
				3	
		106	-DJZ	4	
		1G7	-DJNZ	4	
		122	-MDLC	3	
		124	-HESF	1	
		129	-HC	3	
		130	-HRT	2	

NSWC AR/LYR-7 (CP)  
INDEX TO DATA ITEMS

14 DEC 79 PAGE 149.018

INDEX TO DATA ITEMS

PAGE LINE TYPE NAME AND REFERENCES

PAGE	LINE	TYPE	NAME AND REFERENCES
61	131	OC_SORT	3 3 3 4 5 5
61	38	PUT_SRREG	2 3
61	40	FLAGGING_ADD_SUBTRACT_HEADER	2
71	-CASE	_000	_F51 6
71	127	_HCR	3
71	-CLASS	5	INTERRUPT_SCAN
71	6	INTERRUPT_SEQUENCE	6 12 16 17 21 34
12	GET_ISC	4	6 7 8 9
12	74	8	11
74	-IP	5	6 7
71	-CODE	5	INTERRUPT_SCAN
71	11	GENERATE_SYNCHRONUS_INTERRUPT	8 11 17 20 27 34
12	GET_ISC	6	24 25 26 27 27 30 36
12	74	5	12
71	-CL	14	OP_READ
		15	OP_STORE
		19	LA_SEQUENCE
72	-DFCCDE	3	CP MAIN LOOP
72	55	16	20 19

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI		-DISPLACEMENT	23
DI		-DO_JUMP	4
DI		-ENDCASE	
DI	127	-HOR	16
DI	-INTERRUPT		
DI	11	GENERATE_SYNCHRONOUS_INTERRUPT	23
DI	-ISC	8 INTERRUPT_SEQUENCE	5
DI			7 13
EI	-JUMP	114 -J531	5
DI		-MASK	14
DI		-OP_READ	6
DI		-OP_STORE	13
DI		-IA_SEQUENCE	13
DI		-PATCH	4
DI	115	-J532	11
DI	116	J533	2
DI			12 16 20 22 24
DI	-NDRU	19 MEMORY READ	8
CI	-NLP	53 -LLP	6
DI	-P	14 OP_READ	13
DI		-OP_STORE	6
DI			13

NS/C AN/URK-7 (CP)  
INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

PAGE	LINE	TYPE	NAME AND REFERENCES
DI			18 IA_SEQUENCE 4 36
DI	_RNLP	53	_LLP 6
DI	_STEP	3	CP MAIN LOOP 2 17 BPR_CHECK 30
DI	_WHILE	54	_CNT 4 105 _JEP 10
DI	_XRL	55	_XR 11 14

NSWC

AN/UYK-7 (CP)

14 DEC 79 PAGE 150

\*\*\*\*\*  
\* INDEX TO FLOW SEGMENTS \*  
\*\*\*\*\*

INDEX TO FLOW SEGMENTS

PAGE LINE TYPE NAME AND REFERENCES

32 FS ADD\_S 4 I-SEQUENCE

7  
13 JUMP\_ADDRESS  
16  
14 OP\_READ  
20  
15 OP\_STORE  
22  
18 IA\_SEQUENCE  
11  
55 -XR 7  
73 -10 12  
1111 -JS 5  
112 -JL 10

17 FS BPR\_CHECK 4 I-SEQUENCE

9  
13 JUMP\_ADDRESS  
18  
14 OP\_READ  
21  
15 OP\_STORE  
23  
16 IA\_SEQUENCE  
12  
55 -XR 8  
1111 -JS 6  
112 -JL 11

104 FS CG  
22 FS CP/ICC\_CLOCK\_COMMUNICATIONS  
137 -HSIM 5  
138 -HSTIC 1

NSWC ANUYK-7 (CP)  
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES		
3	-	FS	CP MAIN LOOP		
62	FS	DA			
45	FS	DIVIDE_COMPARE	68	_FD	15 19
			97	DO_DIVIDE	15
97	FS	DO_DIVIDE	98	_D	5
			129	_H0	5
23	FS	DO_JUMP	105	_JEP	16
			106	_DJZ	7
			107	_DJNZ	7
			108	_F51	13
			109	_LBJ	10
			110	_JBNZ	8
			113	_J530	32
			114	_J531	7
			115	_J532	27
			116	J533	23
96	FS	DO_MULTIPLY	67	_FM	14
			95	_H	5
			128	_HM	5

NSC ANJUYK-7 (CP)  
INDEX TC FLOW SEGMENTS

INDEX TC FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
131	FS	DC_SORT	13C _HRT 11 14
40	FS	FLOATING_ADD_SUBTRACT_HEADER	
42	FS	FLOATING_NORMALIZE	65 _FA
			66 _FAN 7
			66 _FD 7
41	FS	FLOATING_OVERFLOW	*3 FLOATING_ROUND
			65 _FA 10
			66 _FAN 5
			66 _FD 5
			68 _FD 27 32
46	FS	FLOATING_POINT_END	65 _FA
			66 _FAN 12
			67 _FM 12
			67 _FM 26
43	FS	FLOATING_ROUND	
			65 _FA 10
			66 _FAN 10
			67 _FM 21
11	FS	GENERATE_SYNCHRONOUS_INTERRUPT	
			17 BPF_CHECK 39 42
			18 IA_SEQUENCE 30 32
			19 MEMORY_READ 11
			20 SPP_CHECK 18 24 30 34

14 DEC 79 PAGE 150-004

ANSWER-7 (CP)  
INDEX TC FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES	
			46	FLOATING_POINT_END 10
	60	-TSF	2	
	64	-LMP	2	
	66	-FD	6	11
	69	-XS	6	
	70	-IPI	2	
	71	-AEI	2	
	72	-LIP	2	14
	73	-IG	2	14
	74	-IK	2	17
	112	-JL	2	
	115	-J532	17	
	116	J533	5	
	117	LCT	6	
	118	SCT	7	
	137	-HSIM	7	
	139	-HPI	2	
	140	-H776	2	
33	FS	GET_AREG	24	GET_SHIFT_AMOUNT 11
			40	FLOATING_ADD_SUBTRACT_HEADER 8 9
			48	-CR
			49	-SC
			50	-PS
				3
				2

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	51		-XOR
	52		-ALP
	53		-LLP
	56		-SLP
	57		-SSUP
	58		-SDIF
	59		-DS
	62		-DA
	63		-DC
	65		-FA
	66		-FAN
	67		-FH
	68		-FD
	78		-LDIF
	79		-ANA
	80		-AA
	81		-LSUP
	88		-SA
	90		-SNA
	91		-SH
	93		-RA
	95		-R
	96		-D
	1C1		-C

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES		
			102	-CL	3
	103	-CM	2	3	
	104	-CG	2	3	
	105	-JEP	3	5	
	106	-DJZ	5	6	
	107	-DJNZ	4	5	
	108	-FS1	4	5	
	120	-HLC-61			
	121	-HLC	13		
	122	-HDLC	3		
	123	-HSF	3	4	
	124	-HDSF	1		
	125	-HCP	1	2	
	126	-HDCP	1		
	127	-HCR	1		
	128	-HR	1	2	
	129	-HD	2	3	
	130	-HRI	2	3	4
	133	-HC	3	13	
	134	-HCL	2	3	
	135	-HCP	2	3	5
24	FS	SET-BSEG	24	SET-SHIFT-LAMENT	
				E	
				-LBMP	3

MSW/C

ANUJK-7 (CP)  
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	69	-XS	4
	70	-IPI	4
	71	-AEI	8
	72	-LIM	8
	73	-IO	8
	77	-LXB	9
	84	-LB	4
	85	-AB	3
	86	-ANB	3
	87	-SB	3
	89	-SXB	2
	100	-CXI	4
	110	-JBNZ	1
	111	-JS	2
	132	-SXB	1
	136	-HCB	2 3
12	FS	SET_IS	5 INTERRUPT_SCAN
24	FS	GET_SHIFT_AMOUNT	{ 6 17
		121	-HLC 2
		122	-HOLC 2
35	FS	GET_SREFS	{ 32 ADDS 8

NS=C            AUYK-7 (CP)            INDEX TO FLCW SEGMENTS

14 DEC 79   PAGE 150-008

INDEX TO FLCW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
25	FS		HALF-WORD_TOGGLE
120	FS		HLC_61
119	FS		MSC_60
18	FS	IA_SEQUENCE	13 JUMP_ADDRESS
			14 OP_READ 6
			15 OP_STORE 6
			69 -xs 2
			7C -IPI 6
			71 -AEI 6
			72 -LI* 6
			73 -IO 6
			75 -RF 7
5	FS	INTERRUPT_SCAN	3 CP MAIN LOOP 18
6	FS	INTERRUPT_SEQUENCE	5 INTERRUPT_SCAN 34 11 GENERATE_SYNCHRONOUS_INTERRUPT 46
4	FS	I_SEQUENCE	3 CP MAIN LOOP 15
13	FS	JUMP_ADDRESS	55 -XR 3 105 -JEP 2 106 -DJZ 2

MSYC AM/UYK-7 (CP)  
INDEX TO FLOW SEGMENTS

14 DEC 70 PAGE 150-CG9

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES	
14	107		-DJNZ	2
	108		-FS1	2
	109		-LBJ	8
	110		-JBWZ	6
	112		-JL	4
	113		-JS30	2
	114		-JS31	2
	115		-JS32	3
	116		-JS33	24
	116	FS	J533	2
117	FS	LCT		
14	FS	MEMORY_READ	4 I-SEQUENCE 12	
			13 JUMP_ADDRESS 17	
			14 OP_READ 22	
			15 OP_STORE 24	
			18 IA_SEQUENCE 13	
			55 _XR 12	
			112 _JL 13	
14	FS	OP_READ	4C FLAGGING_ADD_SUBTRACT_HEADER 4B -OP 5 6	
			49 -SC 1	
			50 -45 1	

NSWC AN/UYK-7 (CP)  
INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
	51	-XOR	1
	52	-ALP	1
	53	-LLP	3
	54	-CNT	2
	60	-TSF	4
	61	-DL	2
	62	-DA	2
	63	-DC	3
	64	-LBMP	6
	67	-FH	1
	68	-FD	2
	75	-IA	2
	77	-LXB	1
	78	-LCIF	2
	79	-ANA	1
	80	-AA	1
	81	-LSUM	1
	82	-INA	1
	83	-LM	1
	84	-LB	2
	85	-AB	2
	86	-ANS	2
	87	-SZ	2
	88	-PA	1

NSWC

AN/UYK-7 (CP)  
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150-011

INDEX TO FLOW SEGMENTS

PAGE	LIN#	TYPE	NAME AND REFERENCES
	94		_RI 1
	95		_P 3
	96		_D 2
	100		-CXI 2
	101		1Q1 -C 3
	102		102 -CL 4
	1C3		1C3 -CM 4
	104		104 CG 4
	117		117 LCI 2
	11		
15	FS	DP_STORE	27 -REPLACE 8 11
			56 -SLP 4
			57 -SSUM 10
			58 -SDIF 10
			59 -DS 4 5
			60 -TSF 11
			67 -SB 4
			68 -SA 3
			60 -SNA 4
			91 -SM 5
			115 -J532 23
			118 SCI 12
36	FS	PUT_AREG	28 UPDATE_REPLACE 6

ANSI  
INDEX\_TC FILE SEGMENTS

INDEX\_TC FILE SEGMENTS

DATE	LINE	TYPE	NAME AND PREFERENCES
			40 FLOATING_ACD_SUBTRACT_HEADER
	25		
	46		FLOATING_POINT_END
	52	_LLP	7 E
	54	_CST	12
	54	_CST	10
	57	_LSUM	9
	58	_SDIF	9
	61	_DL	4 5
	62	_DA	17 19
	67	_FM	9 10
	7t	_IA	2
	77	_LXB	3
	78	_LGIF	5
	79	_ANA	5
	80	_AA	5
	81	_LSUM	5
	82	_LNA	3
	83	_LM	5
	95	_H	9 10
	98	_D	6 7
	119	HSC_60	27
	121	_HLC	9
	122	_HOLC	10 11
	123	_HSF	10 12
	124	_HCSF	10 11 14

MSWC

ANUWK-7 (CP)  
INDEX TO FLOW SEGMENTS

14 DEC 79 PAGE 150-013

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES	
			125	-HCP 3
			126	-HDCP 3
			127	-HGR 3
			128	-HM 17
			129	-HD 10 11
			130	-HRT 6 7
			131	-HSIM 16 17
			132	-HSTC 7
			133	-HSTC 3
37	FS	PUT_BREG	77	-LXB 6
			84	-LB 6
			85	-AB 5
			86	-AHB 7
			89	-SAB 7
			100	-CKI 6
			109	-LBJ 7
			110	-JBNZ 5
			132	-HLB 5
38	FS	PUT_SRES		
5	FS	REPEAT_SEQUENCE	3	CP MAIN LOOP 22
26	FS	REPLACE_CHECK UPDATE_REPLACE	7	

NSC

AN/UYA-7 (CP)  
INDEX TG FLOW SEGMENTS

INDEX TG FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES		
44		FS	ROUND_UP	68	-FD 34 39
118	FS	SCT			
29	FS	SET_CD1	101	-C	4
			103	-CM	6
			104	C6	6
			133	-HC	6
			135	-HCM	6
			136	-HCB	6
30	FS	SET_CD2	102	-CL	5
			134	-hit	5
20	FS	SPR_CHECK	4	I_SEQUENCE	
			13	JUMP_ADDRESS	
			15		
			14	OP_PREAD	
			15	OP_STORE	
			21	IA_SEQUENCE	
			55	-AR	9
23	FS	UPDATEA_REPLACE	4		
			47	-SC	4
			50	-S2	5
					6

## INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
80	FS		-AA
85	FS		-AB
71	FS		-AEI
52	FS		-ALP
79	FS		-ANA
86	FS		-AH8
99	FS		-BC
92	FS		-B7
101	FS		-C
102	FS		-CL
103	FS		-CH
54	FS		-CNT
100	FS		-CXI
98	FS		-D
63	FS		-DC
107	FS		-DJNZ
106	FS		-DZ
61	FS		-DL
59	FS		-DS
65	FS		-FA
			14

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
66	FS		-FAN
68	FS		-FD
67	FS		-FH
108	FS		-FS1
133	FS		-HC
136	FS		-HCB
134	FS		-HCL
135	FS		-HCM
125	FS		-HCP
			126 -HDGP 4
129	FS		-HC
126	FS		-HDGP
122	FS		-HDLC
124	FS		-HDSF
132	FS		-HLB
121	FS		-HLC
128	FS		-HR
127	FS		-HOR
139	FS		-HPI
130	FS		-HRT
123	FS		-HSF
137	FS		-HSTN
138	FS		-HSTC

## INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
140		FS	-H776
73		FS	-ID
70		FS	-IPI
74		FS	-IR
110		FS	-JBNZ
105		FS	-JEP
112		FS	-JL
111		FS	-JS
113		=S	-JS30
114		FS	-JS31
115		FS	-JS32
76		FS	-LA
84		FS	-LB
109		FS	-LBW
64		FS	-LBWP
78		FS	-LCIF
72		FS	-LIM
53		FS	-LLP
63		FS	-LM
82		FS	-LNA
81		FS	-LSUM
77		FS	-LXB
95		FS	-P

## INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
50	FS	FS	-MS
48	FS	FS	-DR
53	FS	FS	-RA
27	FS	REPLACE	26 REPLACE_CHECK 5
94	FS	FS	-RI
75	FS	FS	-RP
88	FS	FS	-SA
			89 - SXB 3
87	FS	FS	-SB
49	FS	FS	-SC
56	FS	FS	-SDIF
56	FS	FS	-SLP
91	FS	FS	-SM
90	FS	FS	-SNA
57	FS	FS	-SSUM
84	FS	FS	-SX8
60	FS	FS	-TSF
51	FS	FS	-XDR
55	FS	FS	-XP
69	FS	FS	-XS

DISTRIBUTION

Defense Advanced Research Projects Agency  
1400 Wilson Boulevard  
Arlington, VA 22209  
ATTN: William Carlson

U.S. Naval Electronics Systems Command  
Washington, DC 20360  
ATTN: John Machado (Code 330)

Director  
U.S. Army TRADOC System Analysis Activity  
White Sands Missile Range, NM 88002  
ATTN: ATAA-SL (Technical Library)

Commander  
Naval Ocean Systems Center  
271 Catalina Boulevard  
San Diego, CA 92152  
ATTN: Russ Evers (Code 5200)

Defense Technical Information Center  
Cameron Station  
Alexandria, VA 22314

(12)

Defense Printing Service  
Washington Navy Yard  
Washington, DC 20374

Library of Congress  
Washington, DC 20540  
ATTN: Gift and Exchange Division

(4)

RADC/ISCA  
Griffiss Air Force Base  
Rome, NY 13441  
ATTN: Armand Vito

DMA/STP  
Bldg. 56, US Naval OBS  
Washington, DC 20305  
ATTN: Annette Kryqiel

FCDBSA Dam Neck  
Virginia Beach, VA 23461  
ATTN: Cary D. Upshur (Code 6321)

(2)

FCDBSA Code 6  
San Diego, CA 92147

NAVSEC  
Code 6122  
Department of the Navy  
Washington, DC 20362  
ATTN: Joe Mallonee

NUSC  
Code 313  
New London, CONN 06320  
ATTN: Dr. Charles Arnold

NCR E&M--SD  
16550 W. Bernardo Dr.  
San Diego, CA 92127  
ATTN: Leslie Stevens  
Manager, Product Firmware

NANODATA  
6065 Madra Ave.  
San Diego, CA 92120  
ATTN: Robert C. Boe

NANODATA  
One Computer Park  
Buffalo, NY 14203

Sperry Univac  
Univac Park  
PO Box 3525  
St. Paul, MN 55165  
ATTN: Doug Wiedenman  
M.S. U2S17

SDC  
601 Caroline St.  
Fredericksburg, VA 22401

(2)

Hale Associates Research Corp.  
PO Box D.J.  
Stony Brook, NY 11790  
ATTN: John Hale

USC/Information Sciences Institute  
4676 Admiralty Way  
Marina Del Ray, CA 90291

Defense & Space Systems Group of TRW, Inc.  
One Space Park  
Redondo Beach, CA 90278  
ATTN: Barry Press

Naval Sea Systems Command  
Washington, DC 20362  
ATTN: Lowell Wood

GIDEP Operations Office  
Corona, CA 91720

Local:

E41	
K60	
K61 (Carroll Shelton, Henry Walker)	(2)
K70	
K71	
K74	(40)
N30 (R. Hein)	
N52 (Hubbard)	
X210	(6)
X211	(2)